Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Informática

Ambiente Virtual para Manipulação de uma Célula Robotizada

Daniel Conde Diehl

Tito Livio Lapis

Orientador

Prof. Márcio Serolli Pinho

Porto Alegre, 23 de junho de 2004.

Sumário

Lista de Figuras

1. Introdução

Realidade Virtual (RV) é uma simulação gerada por computador de um ambiente tridimensional na qual o usuário deixa de postar-se em frente a um monitor e passa a interagir **imerso** no ambiente da aplicação.

As técnicas de RV estão se tornando cada vez mais importantes como meio de representação de objetos físicos pelo fato de que com seu uso pode-se ter uma economia de dinheiro e também porque em alguns casos se torna impossível de gerar o ambiente real, como por exemplo, simulações de explosões nucleares, ou até situações que possam por em risco a vida do usuário ou causar danos ao equipamento pela má utilização e ainda pode ser utilizada para facilitar o uso por pessoas deficientes que de outra forma não seria possível [SIL 02].

Acredita-se que a Realidade Virtual teve início nos anos 60, onde, o pioneiro no conceito de RV, é o Dr. Ivan Sutherland, com suas contribuições sobre gráficos em computador e interações imersivas em Harvard e na Universidade de Utah. Dentre seus trabalhos mais significativos destacam-se “Sketchpad--A Man-Machine Graphical Communication System", 1963, "Ten Unsolved Problems in Computer Graphics", 1966 e "A Head-Mounted Three-Dimensional Display" 1968 entre outros. Este último trabalho introduziu o conceito do que hoje é considerado como um capacete de realidade virtual, apresentando o chamado de “Sword of Damocles” (Espada de Damocles) [SUN 03] (Figura 1.1).

Figura 1.1 – Head-mounted display de Sutherland

A RV apesar de ter ainda um longo caminho até eliminar a separação existente entre usuário e máquina, já é capaz de fornecer uma gama poderosa de potencialidades para desenvolvimento de aplicações que facilitam a interação do usuário independente da sua área de atuação. Dentro destas potencialidades, uma das áreas que vem tendo uma crescente expansão é a área de treinamento, em especial no que diz respeito à educação técnica e profissional. Isso ocorre, principalmente, em razão da possibilidade de diminuir os custos e o tempo de treinamento devido a maior disponibilidade de equipamentos de simulação em computador, subtraindo a necessidade de se ter o equipamento real, sendo possível, assim, treinar um número maior de usuários, num espaço de tempo menor e com um menor custo.

O presente trabalho tem como objetivo criar um ambiente imersivo para o treinamento do uso de uma célula robotizada, utilizando-se de recursos como óculos de RV e rastreadores de posição. Como base foi utilizado o sistema "ASIMOV", criado pelo Núcleo de Informática Educacional do SENAI-RS, que se compõe de um simulador de célula robotizada não-imersivo. Acreditou-se que a imersão resolveria alguns problemas de interação, visão em profundidade e de perspectiva existentes no sistema atual, trazendo ainda mais realismo a simulação, tornando, com isto, mais fácil e mais atrativo o aprendizado.

Este novo ambiente funciona de forma que o usuário possa ter um controle melhor do sistema com o uso das técnicas de RV. Para isso utilizou-se óculos de RV e rastreadores de posição a fim de possibilitar ao usuário uma sensação e retorno do sistema próximos do real. Com a utilização dos dispositivos de RV podemos dar ao usuário a capacidade de se movimentar e explorar o ambiente de acordo com suas necessidades.

O trabalho foi estruturado da seguinte maneira: o capítulo 2 apresenta exemplos de aplicações de realidade virtual; no capítulo 3 são apresentados ambientes virtuais de treinamento e as características básicas dos mesmos; no capítulo 4 é apresentado o sistema ASIMOV, citado acima; a seguir é abordado os aspectos de implementação do sistema.

2. Aplicações de Realidade Virtual

Atualmente pode-se utilizar aplicações de RV em vários ramos da sociedade como educação, entretenimento, medicina, engenharia, etc.

Na educação a principal vantagem da RV é a interação que ela provém, despertando um maior interesse do usuário. Um exemplo nessa área é o PaulingWorld, um sistema para visualização e estudo de estruturas moleculares complexas e suas interações [LOF 99]. Na Figura 2.1 pode-se observa um exemplo desta aplicação e na Figura 2.2 podemos ver o modo de interação do usuário com o sistema.

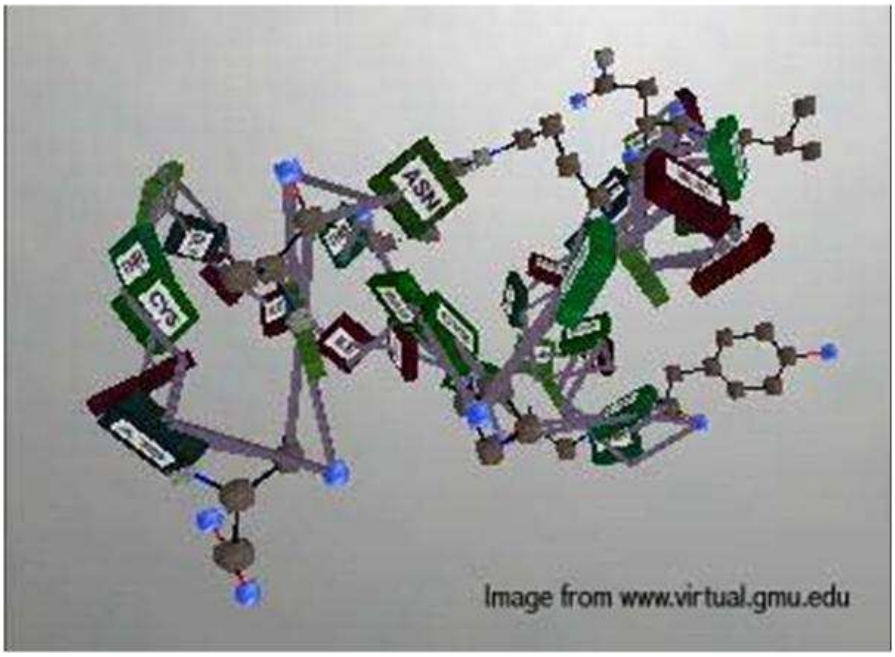


Figura 2.1 – Ambiente *Pauling World*

Figura 2.2 – Usuário interagindo com o *Pauling World*

Na área de entretenimento, temos alguns exemplos de aplicações que usam RV, um deles é o projeto Criação de um Dispositivo de Navegação em Mundos Virtuais (Bicicleta Virtual) [PIN 98]. Este funciona como um ambiente de navegação em que o usuário utiliza uma bicicleta acoplada com sistemas de captura do movimento dos pedais e do guidão, além disto o usuário utiliza um óculos 3D para a visualização do ambiente. Com isto o usuário consegue navegar por uma cidade virtual de forma bastante intuitiva.

Na medicina a realidade virtual é já é utilizada no ensino médico. Um exemplo é o Sistema de RV no Ensino Medico [SAB 99] onde a idéia é permitir aos alunos um passeio virtual altamente realístico, em três dimensões, dentro do ouvido médio de um paciente real. Assim, os médicos poderiam ter uma segurança maior em futuras microcirurgias do ouvido. São feitas imagens extremamente detalhadas de todo o ouvido externo, médio e interno de um paciente real, a partir de tomografia ou raio X. Para poder visualizar em três dimensões de forma real, o usuário utiliza um par de óculos estereocópicos. O componente de interatividade com a imagem é realizado com uma luva (DataGlove), que tem sensores de movimentos em todos os dedos, que são transmitidos para o computador que gera a imagem, de modo que tudo o que as ações da mão do usuário tem repercussão sobre a cena digital mostrada.

Na área de treinamento em engenharia, foco deste trabalho, tem–se muitas vantagens na utilização da RV principalmente em ambientes que possam vir a trazer algum risco ao operador ou que sejam de difícil acesso para treinamento. Em vista disso, muitas empresas se valem desta característica para treinar seus funcionários.

A General Motors, por exemplo, possui um projeto que utiliza um sistema CAVE, denominado VirtualEye, para facilitar o desenvolvimento de novos modelos [EXH 97]. Sua unidade de montagem de caminhões em Detroit e Michigan utiliza *software* da empresa Delmia [DEL 02] para prototipação e avaliação da montagem de seus veículos. A prototipação virtual e as técnicas de manufatura virtual reduzem os custos de desenvolvimento do ferramental empregado, otimizam as operações de manufatura envolvidas e diminuem o *time-to-market* [INT 95].

Como outro exemplo na área de mecânica, tem-se o *software* ASIMOV desenvolvido pelo SENAI-RS, que é um simulador de célula robotizada em um ambiente não-imersivo. O ASIMOV foi desenvolvido com o intuito de facilitar o aprendizado dos alunos que pretendem trabalhar com manipulação de robôs num ambiente de produção.

Segundo Netto [NET 01] o emprego cada vez mais amplo de equipamentos modernos e sofisticados exige mão-de-obra especializada, especialização que questiona os métodos tradicionais de ensino e treinamento, já que demanda a transferência de uma quantidade maior e mais complexa de conhecimento, de uma forma mais eficiente e em um intervalo de tempo menor.

A seguir, no próximo capítulo, serão mostrados alguns ambientes de treinamento e suas principais características.

3. Ambientes Virtuais de Treinamento

Com a necessidade crescente de aprimoramento das técnicas de treinamento o uso de novas tecnologias, como as ferramentas de treinamento, começa a ser visto como uma opção eficaz e viável para suprir as necessidades da área.

A RV é uma tecnologia que tem muito potencial para o setor de treinamento, e, em especial, o treinamento industrial. Assim, as técnicas de RV tornam-se cada vez mais importantes nesse sentido. A primeira razão é que máquinas virtuais têm a vantagem de não precisarem da presença de um *hardware* de custo elevado. Outras vantagens importantes são a potencial redução do tempo de treinamento, maior segurança e transferência de conhecimento através do auto-aprendizado com benefícios nos resultados para o aluno e para a companhia.

Os sistemas virtuais de treinamento se adaptam praticamente a todas as áreas englobando aplicações de diversos níveis de complexidade. A premissa básica em um sistema virtual de treinamento é a **grande possibilidade de interação**. A chave para o sucesso de uma aplicação desse tipo é o feedback real obtido do ambiente e o aprendizado que ele proporciona. Um sistema onde o usuário possa fazer o treinamento diversas vezes, analisando e observando suas respostas, consolida uma base de conhecimento muitas vezes melhor se comparado a um sistema convencional.

A seguir serão apresentados alguns sistemas de diferentes áreas que usam ambientes virtuais de treinamento. Como seqüência serão apresentadas algumas características comuns destes sistemas.

3.1 Exemplos de Sistemas

Um primeiro exemplo de Ambiente Virtual de Treinamento (AVT) são os *trainingpits* ou "cabines virtuais de treinamento". A partir do aparecimento da primeira aeronave, para uso comercial ou militar, houve a necessidade de se treinar e capacitar os futuros pilotos na manipulação dos controles e dispositivos que envolviam estas máquinas. Os primeiros simuladores de vôo tentavam reproduzir o movimento de aeronaves respondendo aos movimentos dos controles mecânicos através de mecanismos com acionamento pneumáticos. Mas nem somente a sensação de se estar dentro de um *cockpit* e o domínio de um simulador de vôo são suficientes para que um

futuro piloto possa estar apto a pilotar uma aeronave. Conhecimentos prévios de todos os comandos, painéis e procedimentos corretos são exigidos destes profissionais. Treinamentos como estes são realizados por poucas empresas especializadas no mundo. No Brasil, a Ground School, com sede em Itajaí – Santa Catarina, é um exemplo de empresa que vem desenvolvendo projetos deste tipo (Figura 3.1).

Figura 3.1 -Cockpit de treinamento

A proposta da empresa é desenvolver um simulador totalmente virtual, que utilizando RV imersiva, (Capacetes e Luvas) possa transmitir ao aluno piloto a sensação real de uma cabine de uma aeronave e realizar seu treinamento, seja de um Boeing ou de um bimotor Brasília, com o auxilio de um computador pessoal em substituição às maquetes físicas atuais. (Figura 3.2) [GAR 02].

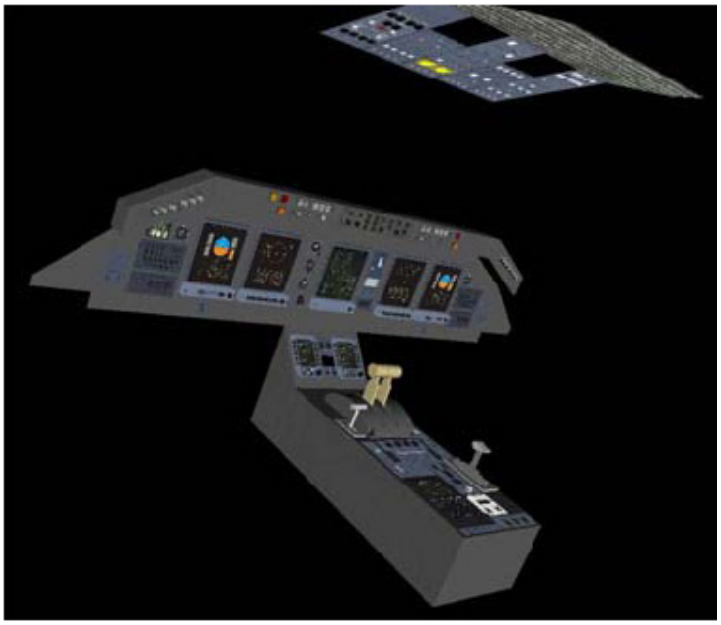


Figura 3.2 - Cockpit de treinamento modelado em 3D

Na área médica a realidade virtual tem atraído um número cada vez maior de simpatizantes, que vêem nesta tecnologia uma ferramenta que possibilita atingir um grau de simulação bastante aceitável principalmente na área de treinamento de procedimentos médicos. Atualmente os novos médicos são treinados com o uso de cobaias e com observação do modo de atuação de médicos veteranos, entretanto acredita-se que seja possível atingir um grau de realismo muito maior com o uso de RV do que com os métodos atuais. Outra vantagem é o monitoramento que pode ser feito a partir da simulação, podendo-se com isto ter uma resposta mais fiel sobre o treinamento realizado pelo usuário, gerando avaliações mais precisas e com isto trazendo uma melhora no aprendizado.

Um exemplo de aplicação é o projeto de treinamento para transplante da medula óssea, onde, sua extração é feita, geralmente, da crista do osso ilíaco com o auxílio de uma seringa. Apesar de aparentemente simples, a coleta exige destreza, pois o paciente doador atravessa um período de preparação que limita seu sistema imunológico. Isto impede que uma coleta mal sucedida possa ser repetida num período curto de tempo. Além disso, o processo de coleta é doloroso em crianças devido à baixa dosagem de anestesia ministrada. No projeto "Sistema de Realidade Virtual para Treinamento em Oncologia Pediátrica" é criado um ambiente virtual para visualização, exame de toque e coleta, onde é possível sentir e tocar o modelo da região pélvica de uma criança através de uma agulha virtual como mostra a Figura 3.3. Para utilizar o sistema o usuário manipula uma agulha real presa a um equipamento *Phantom* capaz de gerar retorno de força sobre a agulha.

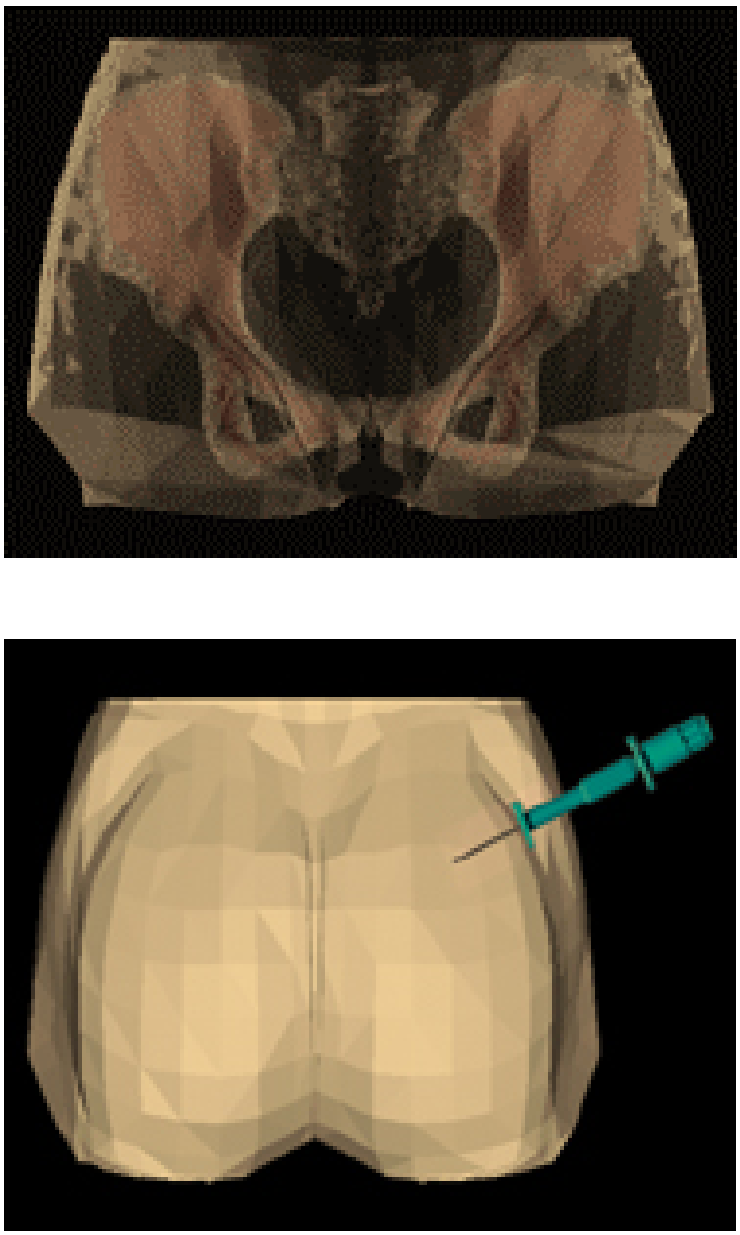


Figura 3.3 - Imagens do Sistema de Realidade Virtual para Treinamento em Oncologia Pediátrica

Na área de treinamento em informática, um exemplo é o projeto COVET, [HOS 96] que permite ao usuário executar um exercício simples de forma colaborativa, isto é, o exercício envolve um usuário (instrutor) que demonstra a outro (aluno) como substituir um cartão defeituoso em um

switch de uma rede ATM. O instrutor demonstra os procedimentos de segurança antes do exercício, a maneira correta em que um cartão defeituoso é removido e o procedimento correto para instalar um novo. As ações dos usuários são comunicadas a outros participantes para que eles possam ter a impressão do envolvimento em um exercício do treinamento como mostra a Figura 3.4.

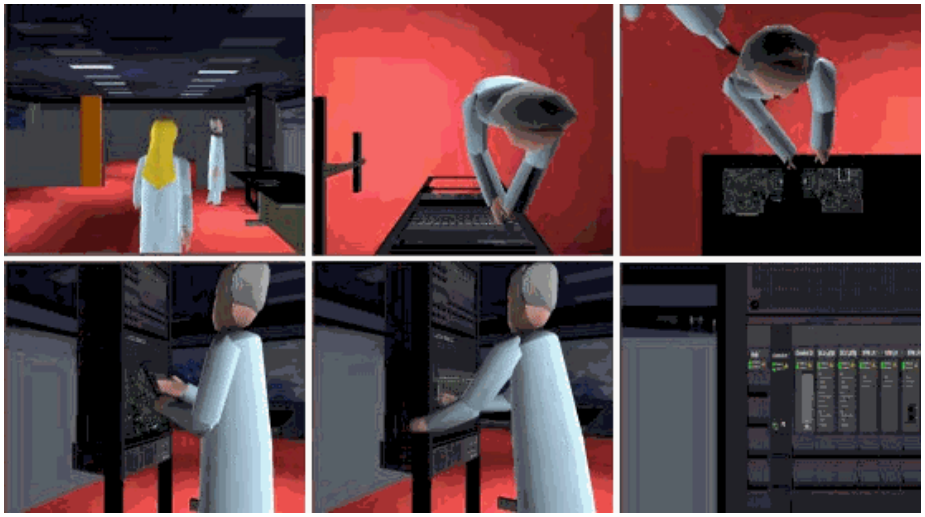


Figura 3.4 - Cenário de treinamento do projeto COVET

Um projeto de treinamento típico da área de engenharia é a implementação de um protótipo de um torno CNC (Comando Numérico Computadorizado) utilizando realidade virtual. Enfocando principalmente, seu sistema de intertravamento e seu modelo geométrico. O intertravamento proposto é acionado à medida que for utilizando o equipamento, isto é, ao se abrir à porta do torno CNC durante uma usinagem, por exemplo. Isto permite ao usuário verificar qual o procedimento adotado pela máquina de acordo com a falha que ocasionou o erro. Muitos intertravamentos são acionados enquanto uma peça está sendo torneada, dessa forma, o projeto possui também a visualização e interação do funcionamento deste processo, bem como a troca das ferramentas existentes no porta-ferramentas e a animação do surgimento e retirada de cavacos, e do líquido refrigerante. Porém existem intertravamentos para aquelas situações onde as falhas ocorrem devido a problemas de *hardware*, como é o caso das falhas do freio do eixo-árvore, do sistema de alimentação de energia elétrica e dos sistemas hidráulico e pneumático do torno. Na (Figura 3.6) temos a visão interna do torno [NET 01].

Figura 3.5 - Vista frontal e traseira do torno CNC com os botões do gerador de falha

Figura 3.6 -Torno CNC com a porta de segurança aberta e com sua vista interna.

Como continuidade estarão sendo apresentadas algumas características comuns a vários sistemas que implementam ambientes de realidade virtual para treinamento.

3.2 Principais Características dos Ambientes Virtuais de Treinamento

Existem várias características comuns entre os Ambientes Virtuais de Treinamento (AVT) apresentados acima. Esta propõe-se a analisar e exemplificar algumas consideradas de maior relevância e que foram freqüentemente encontradas nos sistemas analisados.

A primeira característica relevante encontrada na maioria dos sistemas é a que se refere à questão das possibilidades de **danos** tanto aos equipamentos quanto à integridade física do usuário e das pessoas que estariam envolvidas na atividade caso ela fosse feita em um ambiente real. O protótipo do *torno CNC* é um exemplo onde existe risco de dano físico para a pessoa que está manipulando o torno real. Da mesma forma podemos dizer que os *trainingpits* simulam um ambiente real onde o treinamento, se feito num avião real, traria riscos para o piloto principiante, para o instrutor e para as pessoas situadas próximas a aeronave. O sistema para treinamento em Oncologia Pediátrica pode trazer sérios danos a saúde do paciente em caso de insucesso no procedimento. Nestes casos os AVTs extinguem praticamente este risco visto que tudo se trata de um ambiente virtual não trazendo risco real ao usuário.

Há também a questão de redução do **custo** envolvido no fracasso na execução de uma tarefa que pode trazer grandes perdas em se tratando do modelo real. Um exemplo de AVT com esta característica é o sistema *COVET* onde um procedimento mal sucedido no mundo real pode trazer sérios prejuízos econômicos.

Do ponto de vista da **escalabilidade,** um sistema de treinamento virtual pode atingir um maior número de usuários treinados em um tempo menor, como também, é possível projetar que um único usuário treinado possa fazer a tarefa de vários com o auxílio de uma ferramenta de treinamento como o *COVET,* que, assim como ele simula um treinamento para troca de um cartão num *switch*, ele pode facilmente servir como uma espécie de *help on-line* para as tarefas do usuário.

A **necessidade de imersão** é um item que deve ser avaliado com muito cuidado na elaboração de um projeto de RV para treinamento. Cada sistema deverá ser analisado dentro de suas peculiaridades e detalhes para se decidir há a necessidade ou não da implementação da imersão. Os *trainingpits* são exemplos clássicos da necessidade sensação de imersão no ambiente. Ao contrário, o torno CNC não tem esta necessidade como um fator fundamental. Se analisarmos a parte funcional do sistema.

Um outro ponto importante de se observar no momento da construção de um sistema virtual de treinamento é a questão dos **dispositivos** que serão utilizados. Os dispositivos de E/S devem ser bem dimensionados de modo que o usuário não fique tão distante dos comandos que ele terá no ambiente real. Segundo [NET 01] um ponto negativo em seu projeto foi a ausência de uma segunda mão, pois para trabalhar numa máquina-ferramenta, tendo apenas o mouse para manipular os botões e o tarugo, é uma condição limitante.

O **grau de realismo** com que o sistema deve refletir o real depende fundamentalmente da área da aplicação. O sistema de Oncologia Pediátrica nos remete a um exemplo onde é necessário um grau de realismo extremamente elevado no sentido do retorno tátil. Os *trainingpits* também requerem retorno tátil, mas em um nível diferenciado em relação à manipulação de uma seringa onde precisamos de um sentido de tato bastante detalhado enquanto que os *trainingpits* requerem um módulo de tato não tão elaborado.

A possibilidade de se fazer à **avaliação do usuário** no momento ou após o treinamento é uma outra característica que deve ser considerada. Sempre que for necessário que sejam feitas avaliações sobre a eficácia do treinamento. Com a possibilidade de registro do desempenho do usuário é possível obter resultados mais precisos e mais rápidos das avaliações do usuário. Em sistemas de treinamento de um aspirante a torneiro temos um exemplo onde essa característica é necessária.

4. Células Robotizadas e o Sistema de Treinamento ASIMOV

O sistema ***ASIMOV Simulador de Robótica*** foi desenvolvido no SENAI-RS pela Gerência de Desenvolvimento Educacional – Núcleo de Informática Educacional (NIEd), dentro de um projeto de Pesquisa e Desenvolvimento em Informática Educacional, visando produzir um *software* educacional através da simplificação, simulação ou emulação das tecnologias de produção.

Os esforços para desenvolvimento deste e de outros *softwares* criados no NIEd estão concentrados na idéia de simulação das tecnologias de produção, onde se enquadra o Simulador de Robótica, pois permite aos alunos vivenciarem experiências que se situam entre a manipulação abstrata de conceitos e estratégias e a manipulação direta dos elementos com que se defrontam na prática, por exemplo, montagem e programação de células de fabricação, criando um ambiente que facilita a articulação entre o conceitual e o concreto dos alunos. Outro ponto importante é que a ênfase didática está na possibilidade de reflexão dos conceitos e estratégias sobre uma ação pelos alunos, realizada através da comparação das células de produção com os elementos disponíveis no simulador e dos resultados apresentados na execução do processo pelo computador. Caso o resultado não esteja certo, o aluno deverá depurar o seu processo até a sua correta execução, isto é, depuração do seu próprio conhecimento [ASI 99].

As células robotizadas também conhecidas como células de fabricação, células de produção, ou, ainda, células de trabalho são ambientes que se utilizam de vários elementos ativos ou passivos combinados para a execução de uma determinada tarefa de forma automatizada. Os elementos ativos são aqueles cujas atividades são principais, que manipulam com o produto como, por exemplo, os robôs, os dispositivos CNC (como o torno CNC). Os passivos são aqueles que tem um papel de transporte e armazenamento do produto como as mesas, esteiras e outras ferramentas. As células robotizadas são utilizadas em inúmeras aplicações industriais, entre seus usos podemos citar as fábricas de montagem de automóveis como os robôs de soldagem, a área de embalagem de produtos e etiquetagem, corte de precisão e uma gama interminável de aplicações na área industrial. A vantagem do uso de células robotizadas é o fato de termos uma flexibilização na utilização destes equipamentos.

A proposta do Simulador de Robótica foi enriquecer o aprendizado da programação de robôs, permitindo a montagem e o controle de células robóticas e, principalmente, a programação de robôs através do *Teach Pendant* (uma espécie de controle remoto do robô), de comandos ou de

sua linguagem de programação. O ambiente de simulação do ASIMOV é tridimensional e possui uma série de opções para visualização da célula, possibilitando a programação e o controle do ambiente junto com a visualização dos movimentos dos robôs.

A interface do simulador apresenta três ambientes para interação usuário-computador: Editor de Células, Editor de Programas e Terminal, e mais dois ambientes de visualização: visualização a partir de uma câmera externa e a partir de uma câmera localizada na garra do robô. A Figura 4.1 mostra a interface gráfica básica com suas áreas em destaque que são: (a) barra de título; (b) barra do menu principal; (c) barra de ferramentas para acesso rápido; (d) ambiente de edição ou simulação; (e) editor de célula; (f) visualização; (g) coordenadas do robô.

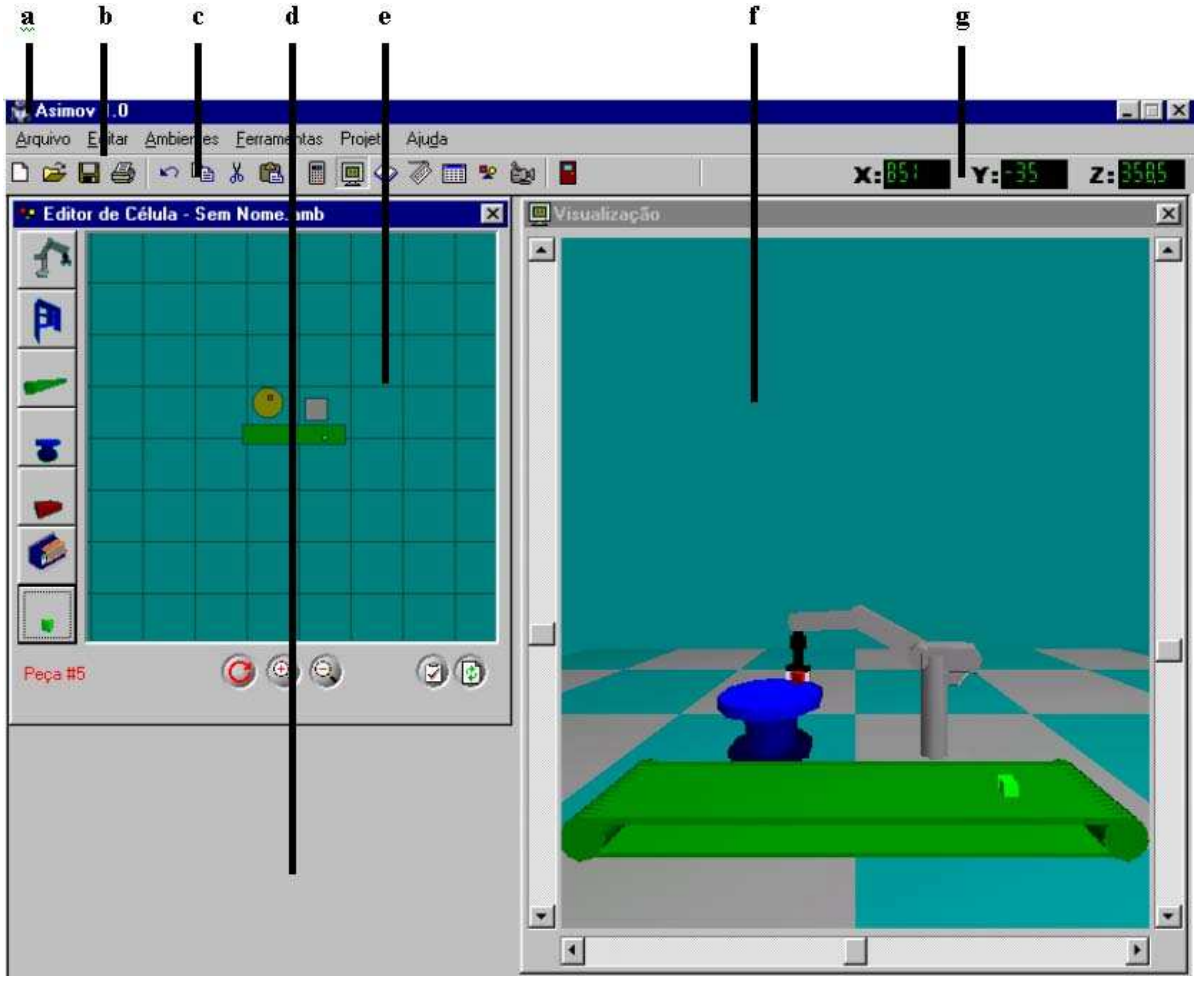


Figura 4.1 - Interface básica do simulador ASIMOV

No modulo **Editor de Células de Trabalho** do simulador é possível criar qualquer célula com os elementos disponíveis no ambiente de edição. Estes elementos servem para interagir com o elemento principal da célula que é o robô ESHED ER-VII. A (Figura 4.2) mostra o ambiente em questão .

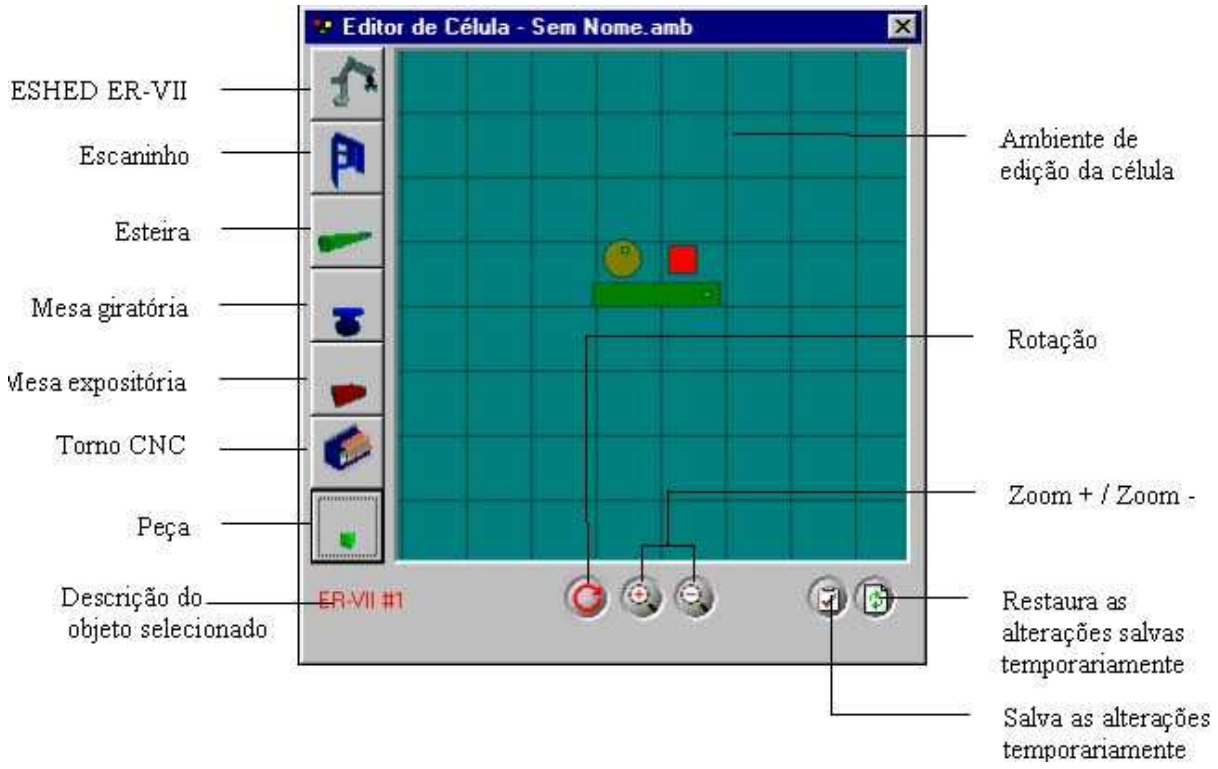


Figura 4.2 - Editor de Célula ASIMOV

No módulo **Editor de Programas** tem-se a disposição uma ferramenta para escrever ou modificar um programa do ambiente Terminal. Esse editor possui os recursos básicos de um editor de texto, como Copiar, Colar, Recortar e Desfazer para edição dos programas. Após a realização das alterações desejadas o programa é enviado para ser o módulo **Terminal** para ser executado. A (Figura 4.3) mostra o ambiente de edição de programas e um exemplo de programa para o robô.

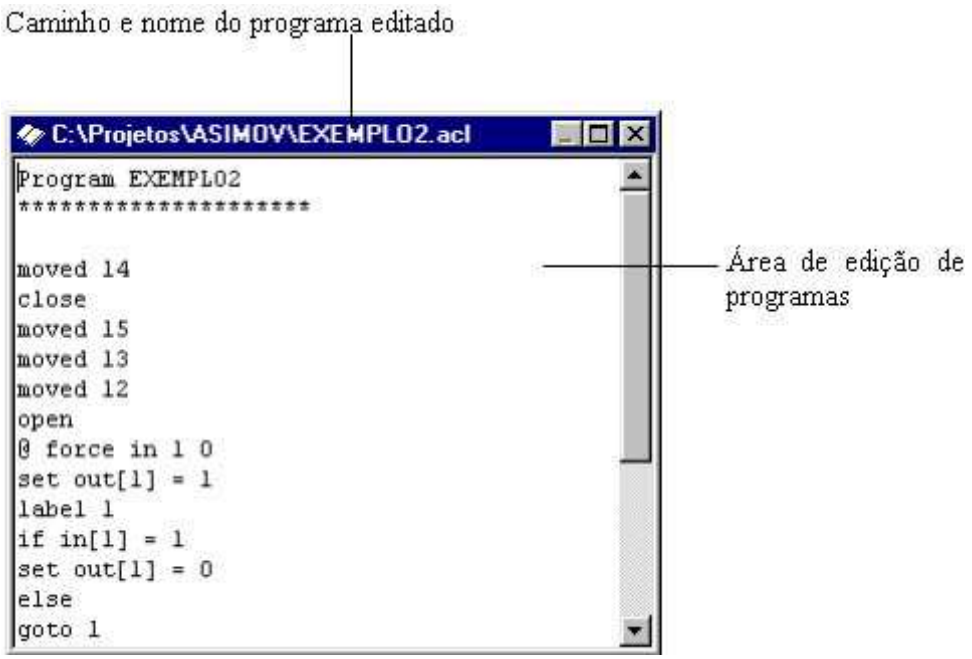


Figura 4.3 – Ambiente de Edição de Programas ASIMOV

No **Terminal** é possível realizar quase todas as funções do robô como editar e executar programas, criar, alterar e remover posições e variáveis, movimentar o robô e seus periféricos, alterar a velocidade, habilitar e desabilitar o controlador. A (Figura 4.4) mostra o Terminal com sua área de entrada de comandos e edição de programas.

Figura 4.4 – Terminal do ASIMOV

O elemento principal de uma célula, que é o robô ESHED ER-VII, pode ser controlado manualmente pelo **Teach Pendant (TP),** que será descrito posteriormente.

4.1 Teach Pendant

O Teach Pendant (TP) é um controlador que auxilia na movimentação do robô Scorbot ER VII. Com ele é possível fazer a movimentação tanto do robô, quanto da mesa e da esteira que formam a célula robotizada. É possível, após a movimentação do robô para um dado ponto efetuar a gravação deste ponto na memória do controlador com o objetivo de utilizá-lo posteriormente na criação de programas para serem executados na célula robotizada.

A movimentação do robô pode ser efetuada através da manipulação da posição de suas juntas (Base, Shoulder e Elbow) ou por coordenadas cartesianas (XYZ). A seleção do modo de movimentação também é feita no TP (Figura 4.6). Os primeiros três botões da esquerda das linhas de botões de movimentação funcionam da seguinte maneira: quando o modo *XYZ* está selecionado os botões superiores incrementam os valores de X, Y e Z da posição da ponta da garra do robô e os inferiores decrementam estes valores. Já quando o modo *Juntas* (JOINTS) está selecionado os botões funcionam controlando o giro da base (BASE), movimentação do ombro(SHOULDER), e movimentação do cotovelo(ELBOW).

Para a implementação do modo *XYZ* foi necessária a utilização de técnicas de **cinemática inversa**, que servem para se descobrir qual o ângulo das juntas é capaz de colocar a ponta da garra em um dado um ponto XYZ no espaço. O uso destas técnicas, está melhor detalhado na seção 6.3. Já para a implementação do modo *Juntas* foi necessária a utilização da **cinemática direta** que trata de descobrir a posição XYZ da garra tendo-se os ângulos das juntas do robô. Estas técnicas , estão descritas na seção 0.

Na garra do robô Scorbot ER-VII tem-se ainda dois outros tipos de movimentação que podem ser controladas pelo TP, que são *Pitch* e *Roll*. O *Pitch* é o movimento de descer e subir da garra, o *Roll* é o giro da garra (Figura 4.5).

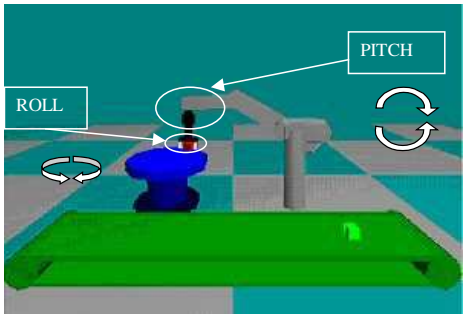


Figura 4.5 – Movimentos *Pitch* e *Roll*

O botão Group Select (à direita, na parte inferior da Figura 4.6) serve para selecionar entre os movimentos do robô (grupo A) e da mesa e da esteira (grupo B).

No visor, é possível visualizar, à direita, o modo de movimentação. Nesta região pode aparecer as palavras XYZ ou JOINTS, no caso da Figura 4.6 está selecionado JOINTS. À esquerda do visor é mostrado o grupo que está selecionado no momento, na Figura 4.6 está selecionado o grupo A. O visor apresenta ainda outras informações referentes à operação que está sendo realizada no momento.

Além do descrito acima o TP ainda possui funções de memorização de pontos, controle de velocidade e movimentação por eixo.

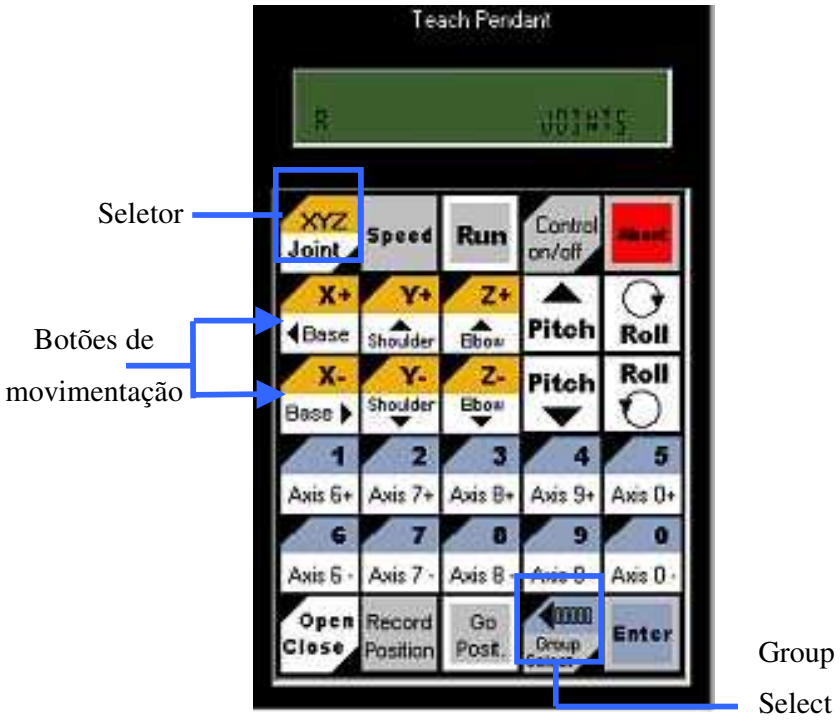


Figura 4.6 - Teach Pendant

Os capítulos a seguir detalham o que este trabalho se propôs a desenvolver e também de que modo isto foi feito.

5. Simulador Imersivo de Célula Robotizada

Neste trabalho foi desenvolvido um ambiente de realidade virtual imersivo para substituir o ASIMOV nas funções de **movimentação**, **visualização** do robô e na **captura** dos pontos de posicionamento da garra para programação do robô.

A idéia de criar um sistema imersivo surgiu da dificuldade imposta pelo ASIMOV para a movimentação dentro da célula robotizada e para a percepção do correto posicionamento dos objetos. Neste tipo de sistema o usuário consegue ter uma melhor noção de onde os objetos realmente estão, fazendo com que o aprendizado seja facilitado, pelo fato de se aproximar mais do ambiente real.

Ressalte-se que o sistema aqui descrito tem integração com o sistema ASIMOV, pois o novo ambiente imersivo serve como uma espécie de sistema intermediário, que recebe o ambiente construído no ASIMOV, permite o posicionamento da garra do robô em um ambiente virtual (usando um *teach pendant* virtual) e possibilita a captura da posição da garra para uso em programas criados no ASIMOV.

A integração entre o ASIMOV e o ambiente virtual foi feita através de um arquivo de dados já existente no ASIMOV e de um relatório de pontos do ambiente que foi criado pelo novo sistema (Figura 5.1). O arquivo de ambiente, gerado pelo ASIMOV, serve de entrada para o sistema imersivo. Ele descreve o posicionamento do robô e dos objetos no ambiente, no caso do robô ainda indicando a posição que se encontram as juntas do mesmo. O arquivo de relatório de pontos gerado pelo sistema imersivo informa os pontos salvos pelo usuário no sistema imersivo para que seja possível o teste de programas utilizando estes pontos, dentro do sistema ASIMOV.

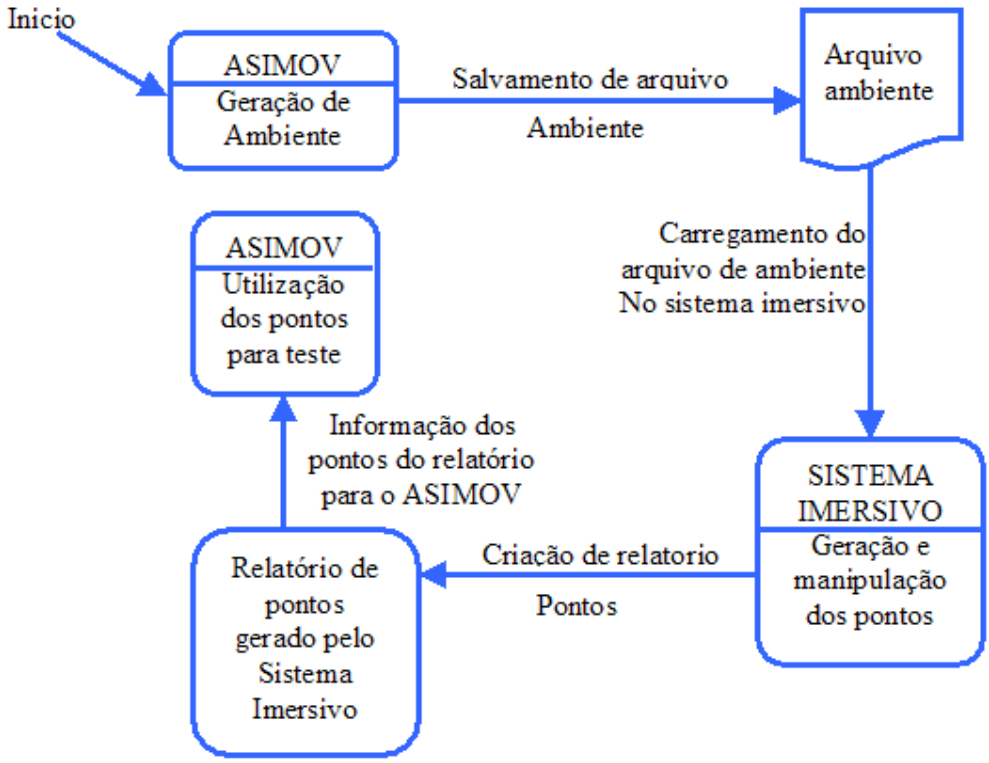


Figura 5.1 -Diagrama de uso dos arquivos

Detalhes do formato do arquivo de ambiente serão apresentados na seção 6.1.

Os equipamentos necessários para o uso do ambiente virtual são óculos de RV e rastreadores de movimento a fim de possibilitar a interação do usuário com o sistema.

O usuário pode visualizar, com o auxílio dos óculos de RV, toda a célula e movimentar-se dentro dela, o que não é possível em uma célula robotizada real devido ao perigo que isso traz à pessoa que entra dentro da célula.

Com um *Teach Pendant (TP)* virtual, o usuário pode mover o robô virtual para que esse interaja com os elementos contidos na célula, através dos comandos já descritos anteriormente. Para a implementação do TP foi utilizada uma técnica de realidade virtual chamada de *pen and tablet* [LIN 99]. Esta técnica consiste em mapear o modelo do TP sobre um objeto real semelhante a uma prancheta que o usuário está manipulando. Na Figura 5.2 e na Figura 5.3 pode-se ver um exemplo de um dispositivo desta categoria sendo usado no projeto Virtual Gorilla Exhibit [BOW 98] para o

apontamento de uma posição em um mapa virtual, para apontamento de objetos e ainda para o acesso aos menus do ambiente.

Figura 5.2 – Visão mundo real interação Pen and Tablet

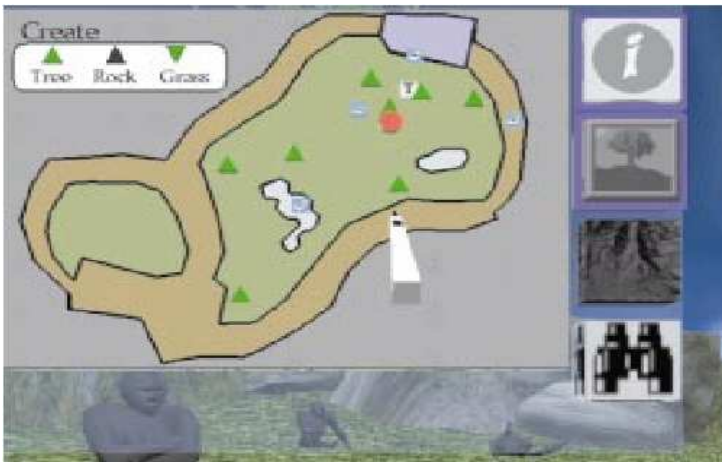


Figura 5.3 – Visão do ambiente virtual interação Pen and Tablet

Outro exemplo de uso da técnica *Pen and Tablet* é o projeto 3D Pallette [BIL 97]. Neste sistema é possível que sejam criadas texturas para objetos que são formadas pela sobreposição de cores através da paleta que fica na mão do usuário (Figura 5.4).

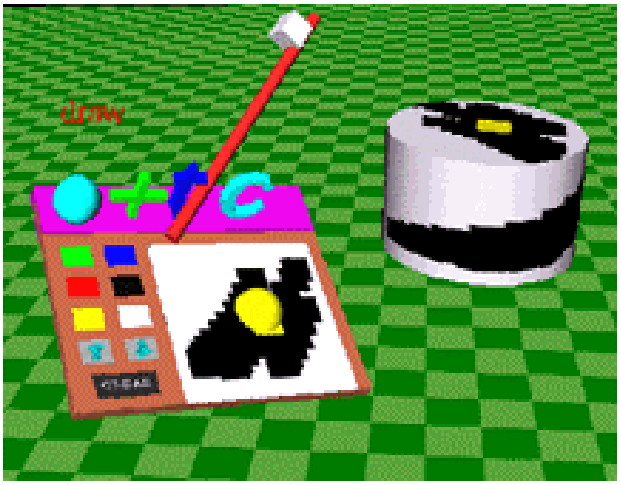


Figura 5.4 - Outro exemplo de pen and tablet chamado 3D pallette.

Para que fosse possível uma sensação de existência do TP real foi fornecido ao usuário um objeto real semelhante a uma prancheta, com as mesmas dimensões do TP, contendo em uma de suas pontas um rastreador de posição. Com isto foi possível exibir a imagem do TP "em cima" da prancheta. Permitindo então a interação com o TP necessária para que seja possível clicar sobre seus botões. Para isto uma espécie de caneta, com outro rastreador de posição, indica em qual botão o usuário está clicando. Assim foi possível ter-se uma interação bastante simples com o TP (Figura 5.5). Os detalhes sobre a implementação do TP são apresentados no capítulo 6.

Figura 5.5 – Teach Pendant Virtual

Para a parte de cinemática inversa foi utilizado o que já está disponível no ASIMOV. Para a cinemática direta foi utilizada a toolkit SmallVR. Os detalhes de como isto foi feito são apresentados nas seções 6.3 e 6.4.

Para a modelagem do robô foi usado um modelo em VRML 1.0 já existente no ASIMOV devidamente transformado para OBJ [WVF 04]. Detalhes sobre a visualização do modelo OBJ do robô são apresentados na seção 6.2.

6. Aspectos de Implementação

Neste capítulo são apresentados os aspectos técnicos das soluções adotadas para os problemas de implementação deste projeto. São tratadas questões como cinemática inversa e cinemática direta, a forma como foram estruturados os arquivos de armazenamento de dados e de intercomunicação entre o sistema proposto e o ASIMOV, como foi modelado o robô e os algoritmos para a implementação do TP.

6.1 Arquivos para troca de dados com o ASIMOV

O ASIMOV utiliza vários arquivos de armazenamento dos dados da simulação, entretanto neste projeto foi utilizado apenas um destes arquivos que é o arquivo de extensão ".AMB", os outros arquivos manipulados pelo ASIMOV não tiveram relevância na construção do sistema imersivo. Na seqüência serão apresentadas descrições sobre os formato desse arquivo.

O arquivo ".AMB" descreve o ambiente gerado pelo ASIMOV que serviu como arquivo de entrada para o ambiente imersivo. Deste arquivo foi lido todo o ambiente previamente montado. Este arquivo é estruturado conforme descrito na Figura 6.1.

O arquivo de ambiente possui em seu conteúdo informações sobre os objetos contidos na célula robotizada, inclusive o robô, informando qual sua posição em X, Y, Z, sua rotação, seu raio(que representa o fator de escala que será aplicado no objeto). Cada objeto possui um ID com o qual ele é identificado no ambiente.

Quando o arquivo é lido se tem uma identificação sobre qual tipo de objeto está sendo referenciado e então este é inserido no ambiente utilizando funções específicas que desenham cada tipo de objeto. Tendo feito isto é aplicada uma translação no objeto para posicioná-lo corretamente, e aplicada uma rotação para deixá-lo na posição correta de acordo com o que estava no arquivo.

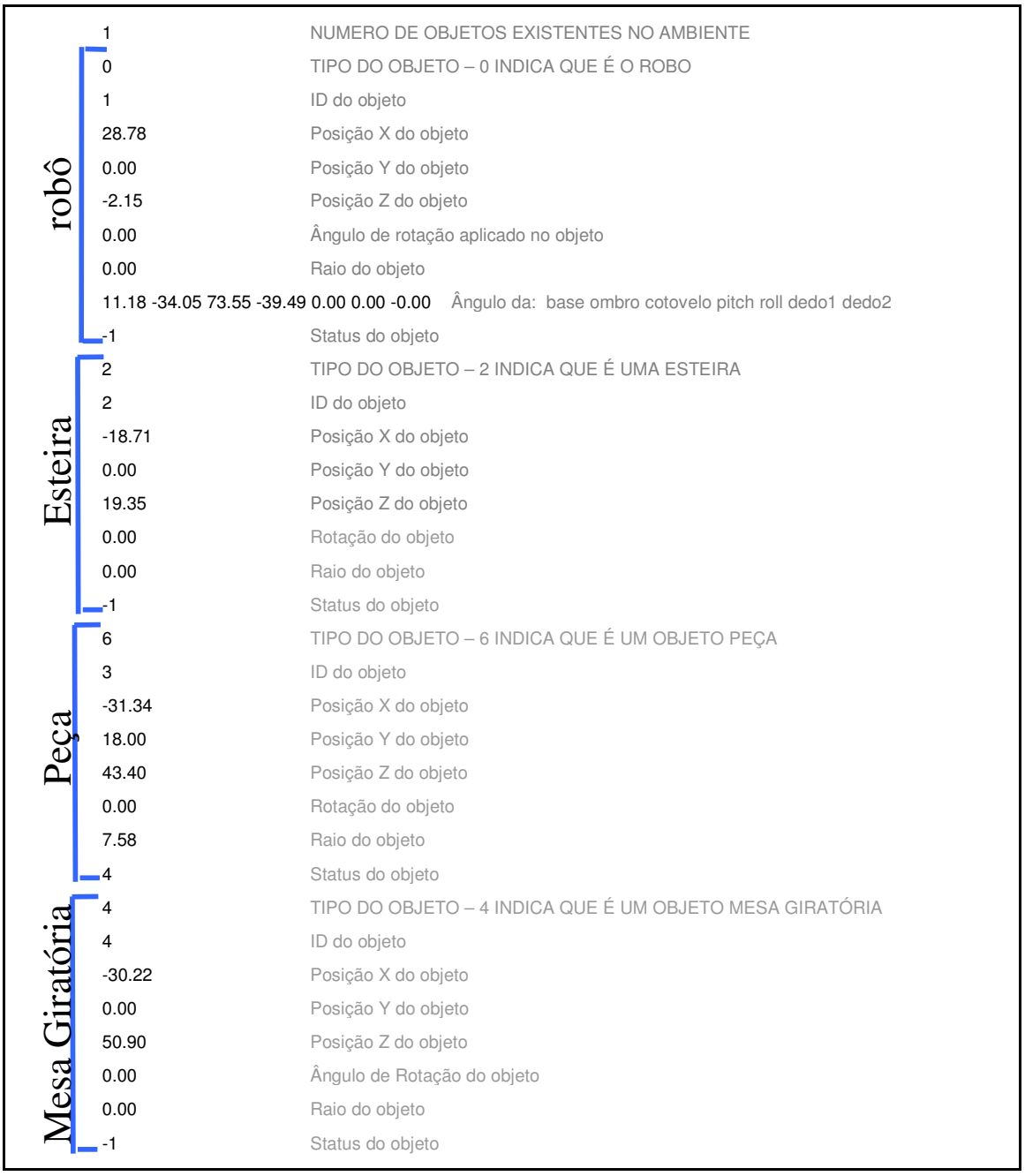


Figura 6.1 - Arquivo .AMB do ASIMOV

Os valores possíveis para o campo “TIPO DO OBJETO” podem ser 1,2,3,4,5 e 6, descritos na que representam os objetos a que serão mostrado logo abaixo,e também é mostrada a função que desenha um paralelepípedo com as dimensões indicadas que está descrita na Figura 6.2. O de tipo igual a 4 representa a mesa de experiências e o tipo igual a 6 representa a peça. respectivamente. O

objeto cujo tipo é 5 representa o torno e é representado por um cubo. Abaixo dos códigos das funções temos o resultado final dos objetos desenhados no ambiente. O desenho dos objetos encontram-se representados na Figura 6.3, Figura 6.4, Figura 6.5, Figura 6.6 e Figura 6.7.

Tipo	Objeto	Figura
1	escaninho	Figura 6.3
2	mesa giratória	Figura 6.4
3	esteira	Figura 6.5
4	mesa de experiências	Figura 6.6
5	Torno*	SEM FIGURA
6	peça	Figura 6.7

*** - Torno não tem representação, por já não existir sua representação no ASIMOV.**

Tabela 6.1 – Objetos da célula robotizada

```
1.      void drawPararelepipedo(int xmin,int xmax,int ymin,int ymax,int zmin,int
        zmax, GLfloat r, GLfloat g, GLfloat b)
2.      {
3.         . . .
4.      }
```

Figura 6.2 – Função para desenho do ambiente

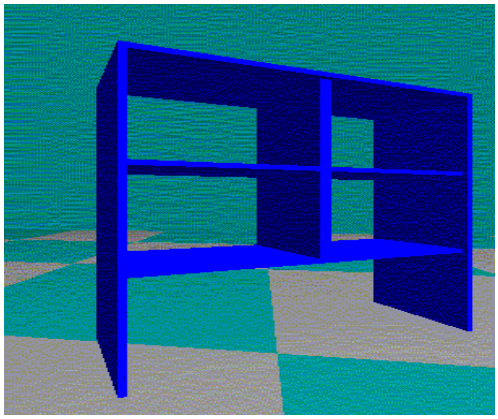


Figura 6.3 – Desenho do escaninho

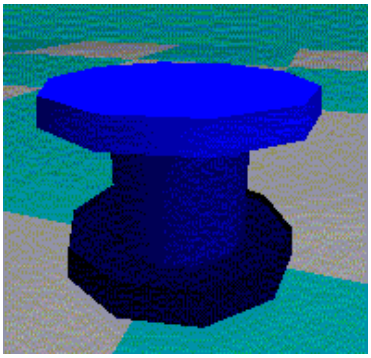


Figura 6.4 – Desenho da mesa giratória

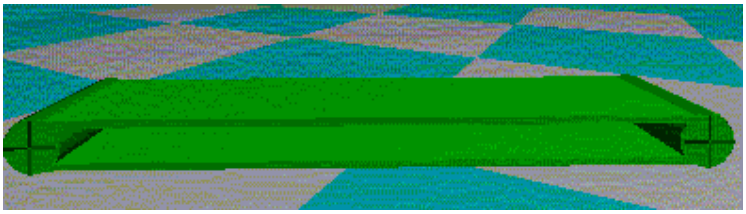


Figura 6.5 – Desenho da esteira

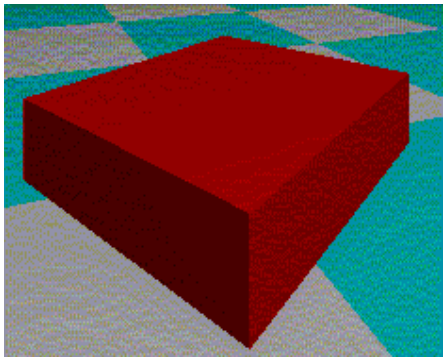


Figura 6.6 – Desenho da mesa de experiências

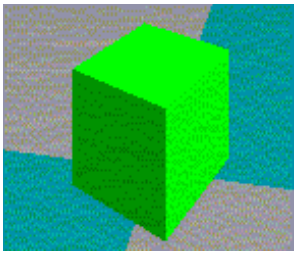


Figura 6.7 – Desenho da peça

O arquivo de saída do sistema imersivo conterá apenas a identificação do ponto que será um contador que é incrementado a cada ponto salvo e as posições X,Y,Z do ponto. Com isto se tem os pontos que o usuário irá testar no sistema ASIMOV, a Figura 6.8 mostra um exemplo de como é este arquivo. Cada linha do arquivo contém um ponto salvo da célula robotizada e sua formatação segue o padrão de: <NRO DO PONTO> = X;Y;Z. Faz-se necessário o uso de ; entre cada ponto para que seja fácil de ser legível para outros programas estes pontos.

```
1. 1 = 851.0;-35.0;358.5
2. 2 = 561.3;-35.0;197.5
3. 3 = 308.6;621.8;564.9
4. 4 = 123.1;-539.1;142.9
```

Figura 6.8 – Exemplo de arquivo de saída do ambiente imersivo

6.2 Modelo tridimensional do Robô

Conforme já foi mencionado, o robô Scorbot ER VII foi modelado no ambiente imersivo a partir de vários arquivos no formato OBJ, onde cada um continha a modelagem de uma parte do robô. Na Figura 6.9 observa-se como estas partes juntas compõem o robô.

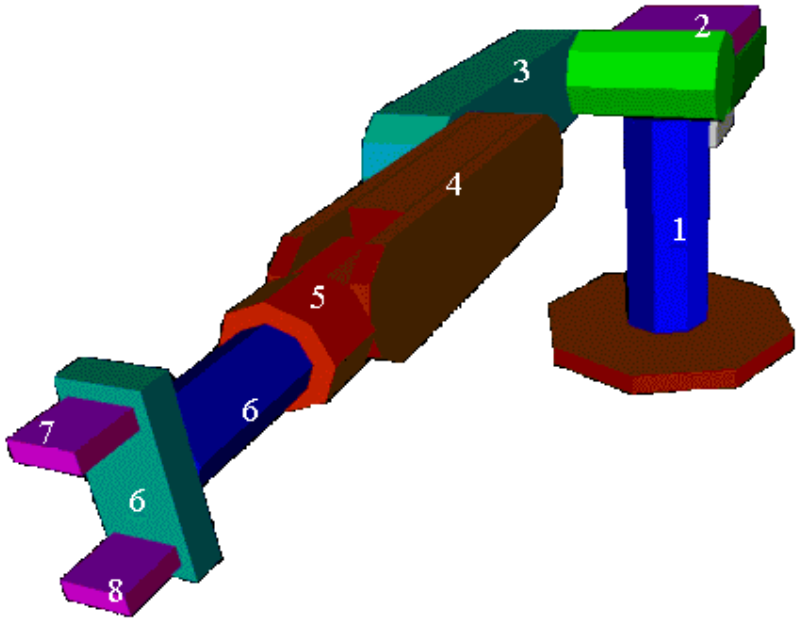


Figura 6.9 - Visualização 3D do robô

Na Figura 6.10 cada uma das partes é apresentada individualmente com seus respectivos nomes.

Além da modelagem geométrica do robô é necessário que se tenha controle dos ângulos das articulações a fim de movimentar o robô. Para tanto, após serem carregados todos os arquivos de objetos foi necessário criar em cada uma das articulações um objeto “pivô” a partir do qual as rotações são aplicadas às juntas. Estes objetos servem como definidores de um novo sistema de coordenadas no qual as rotações são aplicadas.

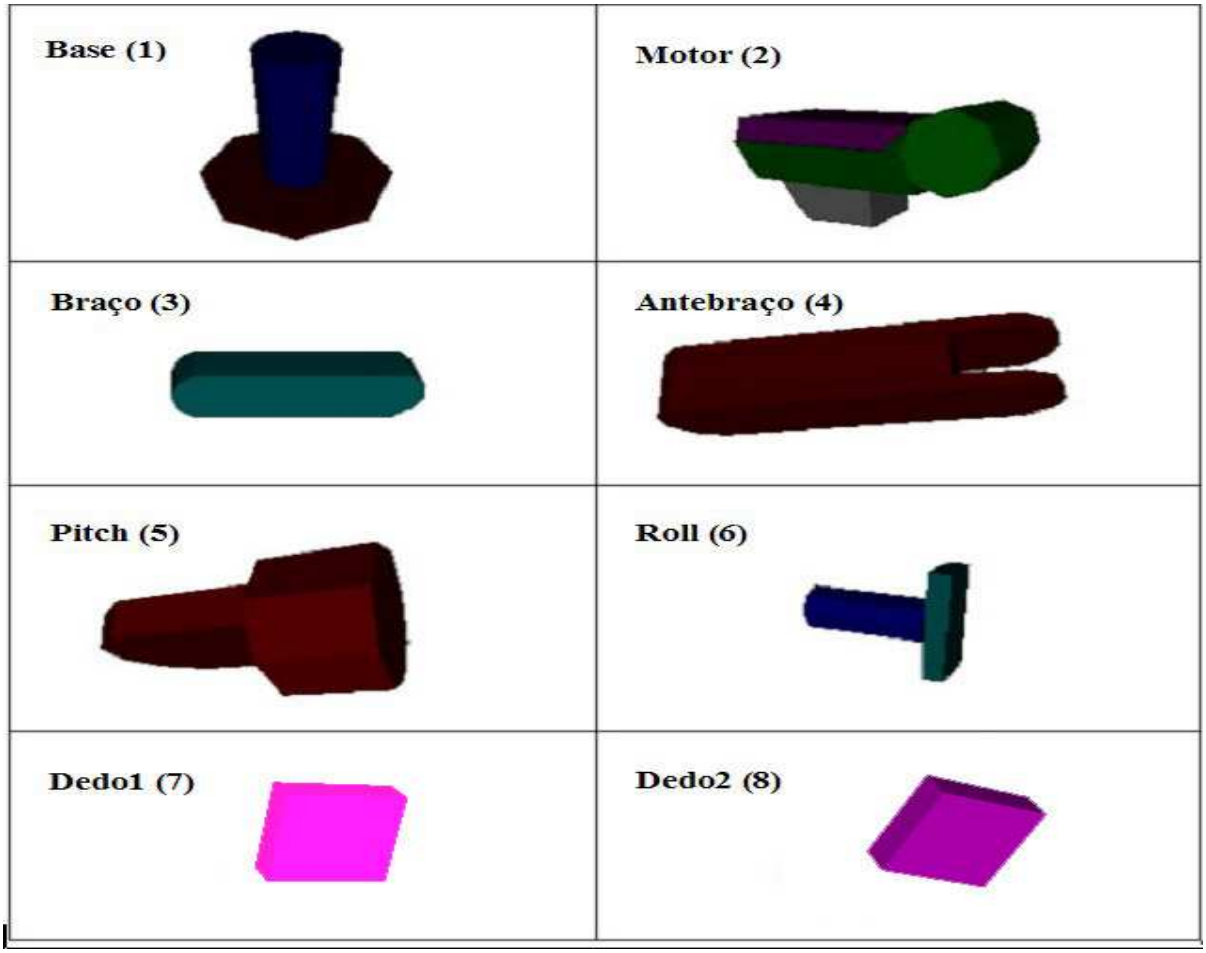


Figura 6.10 – Visualização das partes do robô

Na Figura 6.11, são exibidos estes pivôs, cuja utilização é descrita a seguir:

- A1: Pivô para girar o corpo inteiro do robô no eixo Z;
- A2: Pivô para girar o braço do robô no eixo Y;
- A3: Pivô para girar o antebraço do robô no eixo Y;
- A4: Pivô para girar o pitch do robô no eixo Y;
- A2: Pivô para girar a garra do robô no eixo X;
- D1: Pivô usado para mover o dedo o robô no eixo Z;
- D2: Pivô usado para mover o dedo o robô no eixo Z.

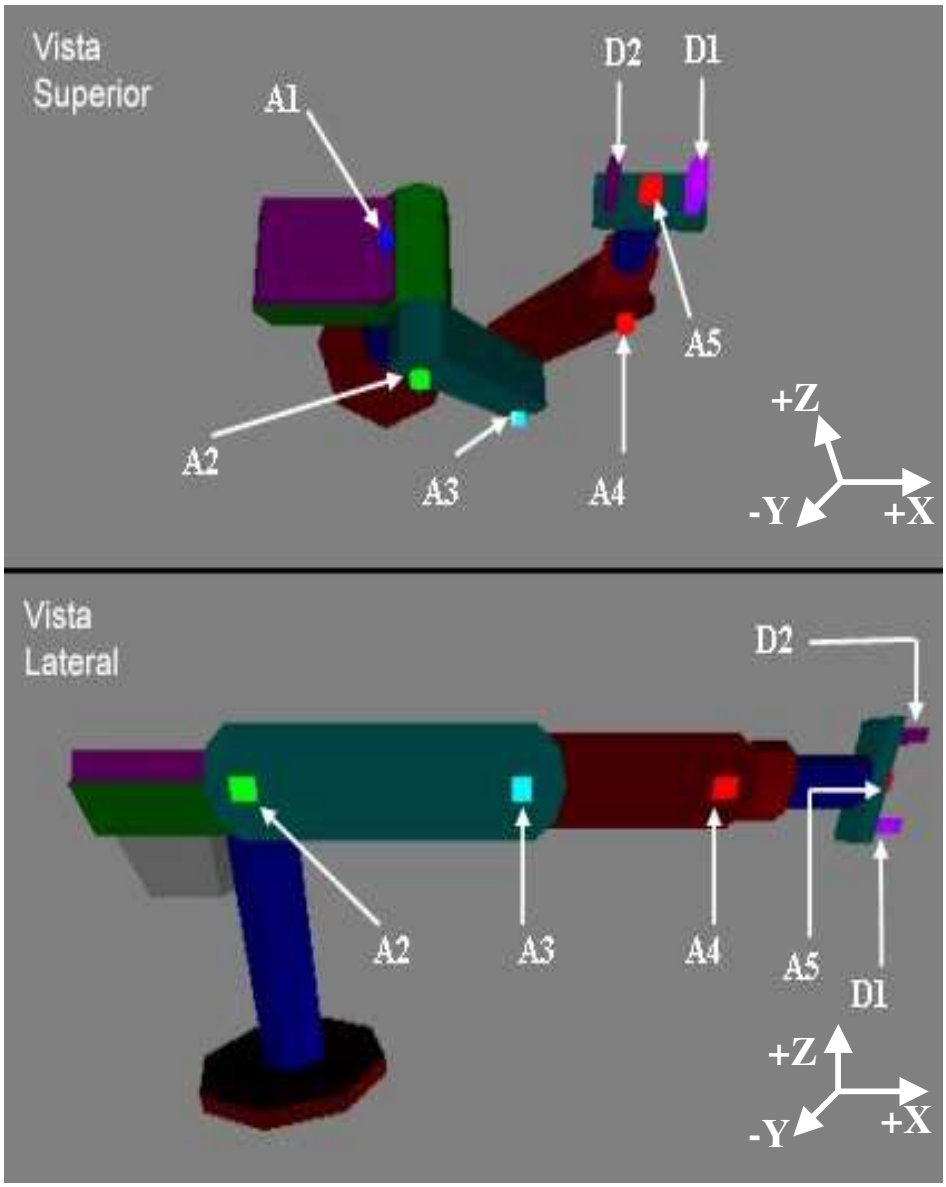


Figura 6.11 – Visualização dos pivôs das juntas

6.3 Cinemática Inversa

Cinemática Inversa é processo de obtenção dos ângulos das juntas de uma seqüência de segmentos articulados a partir da posição do ponto extremo do último segmento da seqüência. No caso de um robô, é o processo de obtenção do ângulo de cada uma de suas juntas, dada a posição

(X, Y, Z) de um ponto localizado no meio de sua garra. Na Figura 6.12 pode-se observar a localização deste ponto. Como no ASIMOV as funções de cinemática inversa já estão implementadas e validadas para o Scorbot ER-VII, estas foram importadas para o ambiente imersivo.

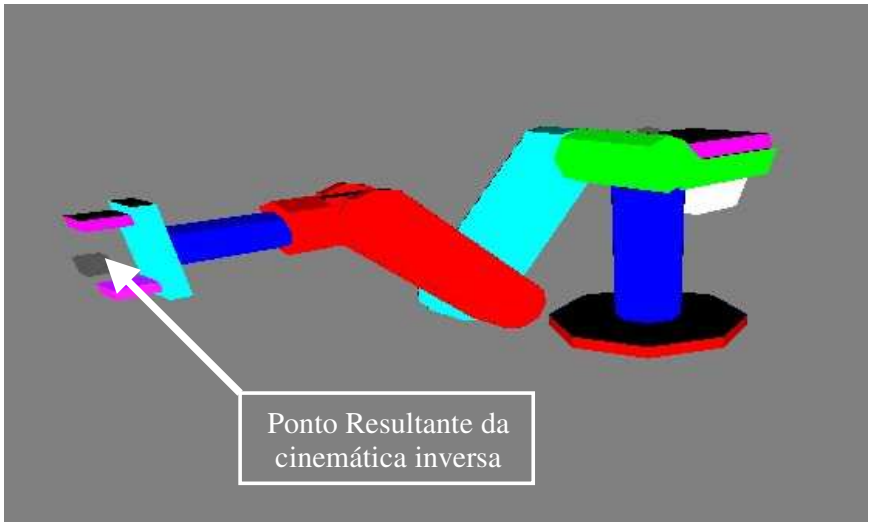


Figura 6.12 – Ponto retornado pela cinemática inversa

Para utilizar a cinemática inversa já implementada no ASIMOV foi necessária uma adaptação do código já implementado para que este passasse a conter apenas a função de cinemática inversa, desprezando as funções de cinemática direta e funções de gráficas de desenho do robô e do ambiente existentes. Além disto foi necessário que esta função fosse compilada como uma DLL e não como um arquivo executável como ocorre no ASIMOV.

A parte do código relativo à cinemática inversa estava implementada em DELPHI e foi compilada na forma de uma DLL (*Dynamic Link Library*), que posteriormente é utilizada dentro do código do ambiente virtual, este código está descrito em C/C++. Nesta DLL foi criada uma função que recebe um ponto relativo à posição desejada para a garra e que retorna os ângulos das juntas corretos para que seja possível mover a garra para alcançar este ponto.

A partir disto o programa em C/C++ que implementa o ambiente imersivo, carrega a DLL e evoca a função de cinemática inversa, informando ponto a ser atingido pela garra e, por fim, chama funções que retornam o valor do ângulo de cada junta. Na Figura 6.13 é apresentado o código de como funciona a integração da DLL com a aplicação em linguagem C, trazendo, logo após, uma breve explicação sobre cada comando contigo no exemplo.

```
1.  //bibliotecas necessárias para o programa
2.  #include <windows.h>
3.  #include <stdio.h>
4.  #include <conio.h>
5.  #include "inversa.h"
6.  //estrutura de retorno da funcao inversa
7.  typedef struct cin{
8.        int status;
9.        double j1,j2,j3,j4,j5;
10. }cinematica;
11.
12. // Declaração de vários tipos pointeiros de funções
13. typedef void (*PROC_INVERSE)(double x,double y,double z,double p,double r, int &l);
14. typedef void (*PROC_GETJ1)(double &ju1);
15. typedef void (*PROC_GETJ2)(double &ju2);
16. typedef void (*PROC_GETJ3)(double &ju3);
17. typedef void (*PROC_GETJ4)(double &ju4);
18. typedef void (*PROC_GETJ5)(double &ju5);
19. //************       FUNCAO DA CINEMATICA INVERSA    ***************
20. cinematica inverse(double x,double y,double z,double p,double r){
21.           //ponteiro para DLL
22.      HINSTANCE ptr;
23.      // Delaração de ponteiros para as funções da DLL
24.      PROC_INVERSE inversa_func;
25.       PROC_GETJ1 getJ1;PROC_GETJ2 getJ2;PROC_GETJ3 getJ3; PROC_GETJ4 getJ4;PROC_GETJ5 getJ5;
26.      Int sucess;   doublé j1,j2,j3,j4,j5;   //variáveis da funcao de conematica inversa
27.      cinematica retorno;
28.      // Carga da DLL
29.      ptr = LoadLibrary("project2.dll");
30.      If (ptr){ // testa para ver se carregou
31.           // "pega" um ponteiro para a função da DLL
32.           inversa_func = (PROC_INVERSE) GetProcAddress(ptr, "inverse");
33.           if (inversa_func){
34.                retorno.status = 1;
35.               (inversa_func)(x,y,z,p,r,sucess);  //execucao da funcao da cinematica inversa
36.               //SE FOR POSSIVEL ALCANCAR DO PONTO INFORMADO
37.               if(sucess){
38.                        // FUNCAO QUE BUSCA O VALOR DA JUNTA 1
39.                   GetJ1 = (PROC_GETJ1) GetProcAddress(ptr, "getJ1");
40.                        if(getJ1)  (getJ1)(j1);
41.                   // FUNCAO QUE BUSCA O VALOR DA JUNTA 2
42.                        getJ2 = (PROC_GETJ2) GetProcAddress(ptr, "getJ2");
43.                   If(getJ2)  (getJ2)(j2);
44.                        // FUNCAO QUE BUSCA O VALOR DA JUNTA 3
45.                   GetJ3 = (PROC_GETJ3) GetProcAddress(ptr, "getJ3");
46.                        if(getJ3)  (getJ3)(j3);
47.
48.                   // FUNCAO QUE BUSCA O VALOR DA JUNTA 4
49.                        getJ4 = (PROC_GETJ4) GetProcAddress(ptr, "getJ4");
50.                   If(getJ4)  (getJ4)(j4);
51.                        // FUNCAO QUE BUSCA O VALOR DA JUNTA 5
52.                   GetJ5 = (PROC_GETJ5) GetProcAddress(ptr, "getJ5");
53.                         if(getJ5)  (getJ5)(j5);
54.              } else  retorno.status = 0;
55.             } else  printf(">>>>>>>Não achou a função <inverse> !!\n");
56.          } else printf(">>>>>>>Erro na Carga da DLL\n");
57.          retorno.j1 = j1; retorno.j2 = j2; retorno.j3 = j3; retorno.j4 = j4; retorno.j5 = j5;
58.          return retorno;
```

Figura 6.13 – código exemplo de utilização de DLL

Neste exemplo assume-se a existência de uma DLL "project2.dll" contendo uma função de cálculo de cinemática inversa ("inverse") e funções de "get" para obter o valor dos ângulos das juntas gerados pelo cálculo. Para a função "inverse" devem ser passados por parâmetro as variáveis *X,Y,Z,Pitch,Roll* e *status*. Neste caso *X,Y,Z,Pitch,Roll* representam o ponto onde se deseja chegar e *status* é um parâmetro de saída, passado por referência, que informa se a cinemática inversa pôde ser calculada com sucesso. Resumidamente os passos para o uso da DLL são os seguintes:

- Incluir a header do Windows de seguinte forma, (ver linha 2):

```
#include <windows.h>
```

- Definir um ponteiro que irá representar a DLL dentro do programa. Está variável materializa a ligação entre a DLL em Delphi e o código em C (ver linha 22):

```
HINSTANCE ptr;
```

- Criar uma variável que receberá um ponteiro para a função que se quer utilizar na DLL (ver linha 24):

```
PROC_INVERSE inversa_func;
```

- Carregar a DLL na memória para que seja possível a utilização de suas funções. Isto é feito com a função "`LoadLibrary`" de maneira que, ao ser carregada a DLL, seja retornado um ponteiro para sua posição de memória (ver linha 29):

```
ptr = LoadLibrary("project2.dll");
```

- Obter um ponteiro para cada função da DLL. Isto é feito com o comando `GetProcAddress`, que retorna um ponteiro para a posição de memória onde está a função da DLL (ver linha 32):

```
inversa_func=(PROC_INVERSE)GetProcAddress(ptr,"inverse");
```

- Fazer a chamada do método de cinemática inversa da DLL a partir do ponteiro previamente obtido, da seguinte maneira (ver linha 35):

```
(inversa_func)(x,y,z,p,r,sucess);
```

Tendo sido apresentado o modo como se carrega uma DLL em C julgou-se relevante a apresentação de como foi montada a estrutura da DLL em Delphi, que é a linguagem em que se encontra o código fonte para a chamada das funções da cinemática inversa do ASIMOV. Na Figura 6.14 é mostrado o código da DLL criada em Delphi. Neste código, destacam-se os seguintes aspectos:

- Indicar que o código em questão é uma DLL com nome Project2 (ver linha 1):

```
library Project2;
```

- Indicar quais bibliotecas serão utilizadas na DLL em questão, para o caso de uma DLL `Sysutil` é necessária sempre (ver linhas 2, 3 e 4):

```
Uses
```

- Declarar o cabeçalho da função `inverse(x,y,z,p,r,&sucesso)`, a palavra Var indica que *`sucesso`* será passado por referência. O que vem logo após `Begin` (linha 9) até o `end` (linha 19) é o código da função em si. Este código apenas declara uma função, posteriormente será mostrado como fazer a função ficar accessível pelos programas que usam a DLL (ver linha 8):

```
procedure inverse(x,y,z,p,r : real;var
sucesso:Integer); cdecl;
```

- Indicar quais as funções poderão ser utilizadas externamente na DLL. Todas as funções especificadas entre o `Export` (linha 60) e o `end` (linha 62) poderão ser chamadas pelos programas que utilizam a DLL. Neste caso as funções são: *inverse, getJ1, getJ2, getJ3, getJ4, getJ5* (ver linhas 59, 60, 61):

```
Exports
   inverse, getJ1, getJ2, getJ3, getJ4, getJ5;
end.
```

```
1.  library Project2;
2.  Uses
3.    SysUtils,Dialogs,uRobot, Classes;
4.  Var
5.    J1,j2,j3,j4,j5:real;
6.    Robot : TRobot;
7.  {$R *.res}
8.  //FUNCAO QUE CALCULA A CINEMATICA INVERSA CONTIDA NA CLASSE TROBOT
9.  procedure inverse(x,y,z,p,r : real;var sucesso:Integer); cdecl;
10. Begin
11.   //CRIACAO DO OBJETO ROBO
12.   Robot := Trobot.Create();
13.
14.   If Robot.Cinematica(j1,j2,j3,j4,j5,x,y,z,p,r) = 1 then
15.   Begin
16.     sucesso := 1;
17.   End
18.     Else
19.     sucesso := 0;
20. end;
21. //-------------------------FUNCOES DE GET DAS JUNTAS------------------------------
22. procedure getJ1(var result:double) ; cdecl;
23. Var
24.  teste:String;
25. Begin
26.   Teste := Copy(floattostr(j1),0,8);
27.   result := strtofloat(teste);
28. end;
29.
30. procedure getJ2(var result:double) ; cdecl;
31. Var
32.  teste:String;
33. Begin
34.   Teste := Copy(floattostr(j2),0,8);
35.   result := strtofloat(teste);
36. end;
37. procedure getJ3(var result:double) ; cdecl;
38. Var
39.  teste:String;
40. Begin
41.   Teste := Copy(floattostr(j3),0,8);
42.   result := strtofloat(teste);
43. end;
44.
45. procedure getJ4(var result:double) ; cdecl;
46. Var
47.  teste:String;
48. Begin
49.   Teste := Copy(floattostr(j4),0,8);
50.   result := strtofloat(teste);
51. end;
52.
53. procedure getJ5(var result:double) ; cdecl;
54. Var
55.  teste:String;
56. Begin
57.   Teste := Copy(floattostr(j5),0,8);
58.   result := strtofloat(teste);
59. end;
60. Exports
61.     inverse, getJ1, getJ2, getJ3, getJ4, getJ5;
62. end.
```

Figura 6.14 – Código delphi da DLL de cinemática inversa

6.4 Cinemática Direta

A Cinemática Direta serve para a obtenção de uma posição (X,Y, Z) da garra a partir dos ângulos das juntas do robô. Esta técnica é utilizada quando se estiver manipulando os valores das juntas pelo TP através do modo *JUNTAS*.

Na implementação de cinemática direta do ASIMOV foi utilizado o algoritmo de Denavit-Hatenberg [HOL 96]. Na criação do ambiente imersivo deste trabalho, entretanto, optou-se por não utilizar este método e sim pela utilização da ferramenta de desenvolvimento SmallVR [PIN 02] que permite, com facilidade a modelagem do robô.

A partir do modelo em OpenGL descrito na seção 6.2, o robô teve sua estrutura organizada em uma hierarquia de objetos utilizando a *toolkit* SmallVR. Nesta *toolkit* foi possível especificar hierarquias de objetos sendo que todo o movimento feito em um objeto-pai é refletido em todos seus filhos, fazendo assim o papel da cinemática direta.

Para isso foi necessário que a modelagem do robô estivesse corretamente estruturada para que não houvesse falha na movimentação. A modelagem na SmallVR foi feita conforme mostra a Figura 6.15. A *base* é o objeto mais alto na hierarquia do robô, logo a seguir como filho da *base* vem o *braço*, depois como filho do *braço* vem o *antebraço*, na seqüência vem como filho do *antebraço* o *punho*, como filho do *punho* há a *mão* que tem como filhos o *dedo 1* e o *dedo 2*. Fazendo isto foi possível mover apenas o pai e a SmallVR automaticamente reflete este movimento em todos seus filhos. Entretanto para que a movimentação do robô funcionasse corretamente foi necessária a inserção dos pivôs, tendo assim todas as aplicações de rotações nos eixos efetuadas sobre os pivôs que se encontram nas junções de cada parte do robô, com exceção de dedo 1 e dedo 2 onde não são necessários pivôs pelo fato de serem efetuadas apenas transformações de translação. A título de exemplo a Figura 6.16 mostra um trecho de código C++ que usa a SmallVR para modelar o robô de sua base até braço.

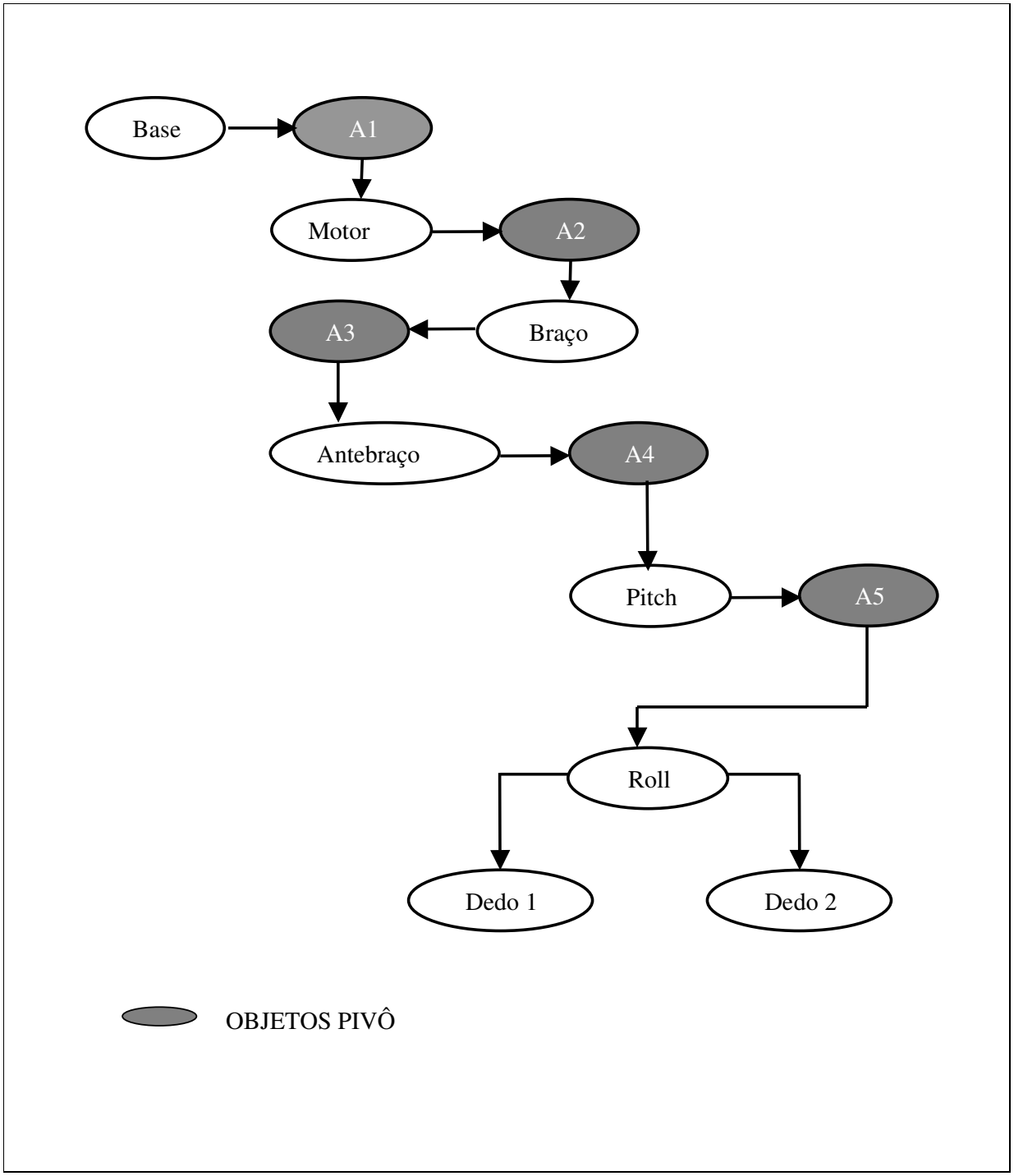


Figura 6.15 – Estrutura hierárquica do robô

```
1.  void monta_robo()
2.  {
3.       //DECLARACAO DOS OBJETOS CARREGADORES DE OBJ
4.       SmVR_CObjFromFile *ObjOBJ = new SmVR_COBJLoader();
5.       SmVR_CObjFromFile *ObjOBJ1 = new SmVR_COBJLoader();
6.       SmVR_CObjFromFile *ObjOBJ2 = new SmVR_COBJLoader();
7.       SmVR_CObjFromFile *ObjOBJ3 = new SmVR_COBJLoader();
8.       SmVR_CObjFromFile *ObjOBJ4 = new SmVR_COBJLoader();
9.
10.
11.      //DECLARACAO DOS OBJETOS DO AMBIENTE
12.      SmVR_CGeometricObject *base, *tronco, *cima_traz, *redondo, *braço;
13.      SmVR_CGeometricObject *a1, *a2;
14.
15.      //MONTA A BASE
16.      if (ObjOBJ->Load ("TestFiles\\Objetos3D\\base.obj")) {
17.           //INSERE OBJETO BASE NO AMBIENTE
18.           base = new SmVR_CgeometricObject ("base", ObjOBJ);
19.           //INSERE BASE NO TOPO DA HIERARQUIA DO ROBO, COMO FILHO DO rootObject
20.           rootObject->AddChild (base);
21.      }
22.
23.      //MONTA O TRONCO QUE É A HASTE DA BASE
24.      if (ObjOBJ1->Load ("TestFiles\\Objetos3D\\tronco.obj")) {
25.           tronco = new SmVR_CGeometricObject ("tronco", ObjOBJ1);
26.           base->AddChild(tronco);
27.      }
28.
29.      //MONTA A PARTE DO MOTOR(PARTE TRASEIRA DO BRACO)
30.      if (ObjOBJ2->Load ("TestFiles\\Objetos3D\\parte3.obj")) {
31.           //INSERE OBJETO REFERENTA AO PIVO 1 NO AMBIENTE
32.           a1 = new SmVR_CGeometricObject ("a1", DrawCubo);
33.           //POSISICIONA O PIVO NA POSIÇÃO CORRETA
34.           a1->TranslateBy(-50,0,400);
35.           //FAZ O PIVO FICAR INVISIVEL
36.           a1->SetVisibility(0);
37.           //FAZ O PIVO SER FILHO DO OBJETO TRONCO
38.           tronco->AddChild (a1);
39.           cima_traz = new SmVR_CGeometricObject ("cima_traz", ObjOBJ2);
40.           //FAZ O OBJETO CIMA_TRZ SER FILHO DO PIVO a1
41.           a1->AddChild (cima_traz);
42.      }
43.      //MONTA CILINDRO REDONDO QUE LIGA MOTOR AO BRACO
44.      if (ObjOBJ3->Load ("TestFiles\\Objetos3D\\redondo.obj"))  {
45.           redondo = new SmVR_CGeometricObject ("redondo", ObjOBJ3);
46.           cima_traz->AddChild(redondo);
47.      }
48.      //MONTA PRIMEIRO SEGMENTO DE BRACO DO ROBO
49.      if (ObjOBJ4->Load ("TestFiles\\Objetos3D\\braco.obj")) {
50.           a2 = new SmVR_CGeometricObject ("a2", DrawCubo);
51.           //POSICIONA O PIVO CORRETAMENTE
52.           a2->TranslateBy(0,-160,355);
53.           //FAZ O PIVO FICAR INVISIVEL
54.           a2->SetVisibility(0);
55.           redondo->AddChild(a2);
56.           braco = new SmVR_CGeometricObject ("braco", ObjOBJ4);
57.           a2->AddChild(braco);
58.      }
59.  }
```

Figura 6.16 – Código SmallVR para montagem do robô

Neste código tem-se a maneira como é carregado o robô para o ambiente virtual a partir dos objetos "OBJ" já existentes. Os objetos são carregados de forma individual e inseridos na estrutura hierárquica do robô, não sendo necessário aplicar nenhuma transformação geométrica nos objetos pelo fato dos mesmos terem sido modelados nas posições e tamanhos corretos.

Nos objetos do tipo pivô, como é o caso do *a1* (ver linha 32 da Figura 6.9), é necessário inseri-lo no ambiente, e posicioná-lo nos locais corretos com translações e depois então fazer com que o mesmo seja filho do objeto anterior da hierarquia e que o objeto posterior da hierarquia seja filho do pivô.

A seguir será apresentada uma breve explicação dos passos feitos no algoritmo da Figura 6.16 para a criação do robô:

- Declarar o objeto que conterá o "OBJ" (ver linha 4):

```
SmVR_CObjFromFile *ObjOBJ = new SmVR_COBJLoader();
```

- Carregar os objetos "OBJ" isto é feito utilizando o código, (ver linha 16):

```
if (ObjOBJ->Load
("TestFiles\\Objetos3D\\base.obj")) {
```

- Declarar a variável que representará o objeto na estrutura hierárquica (ver linha 12):

```
SmVR_CGeometricObject *base
```

- Instanciar o objeto passando como parâmetro o nome que objeto terá no ambiente e a variável que contém o objeto "OBJ". Isto é feito com o seguinte comando (ver linha 18):

```
"base = new SmVR_CGeometricObject ("base",
ObjOBJ);"
```

- Inserir o objeto na estrutura hierárquica do robô (ver linha 20, 38 e 41):

```
rootObject->AddChild (base);
tronco->AddChild(a1);
a1->AddChild(cima_traz);
```

- Posicionar o pivô na sua posição correta (ver linha 34):

```
a1->TranslateBy(-50,0,400);
```

- Tornar o objeto pivô invisível (ver linha 36):

```
a1->SetVisibility(0);
```

6.5 Modelagem geométrica do Teach Pendant

No ambiente virtual, o *teach pendant* TP é representado por um paralelepípedo sobre o qual se mapeou a mesma imagem que existe no teclado do TP real conforme a figura 5.5.

Para permitir que o usuário selecione os comandos do TP virtual, criou-se um *apontador* com o qual ele pode apontar o comando (botão) desejado como se estivesse usando uma caneta (*apontador*). Este apontamento se dá pela colocação da caneta sobre a tecla escolhida, conforme a Figura 6.17.

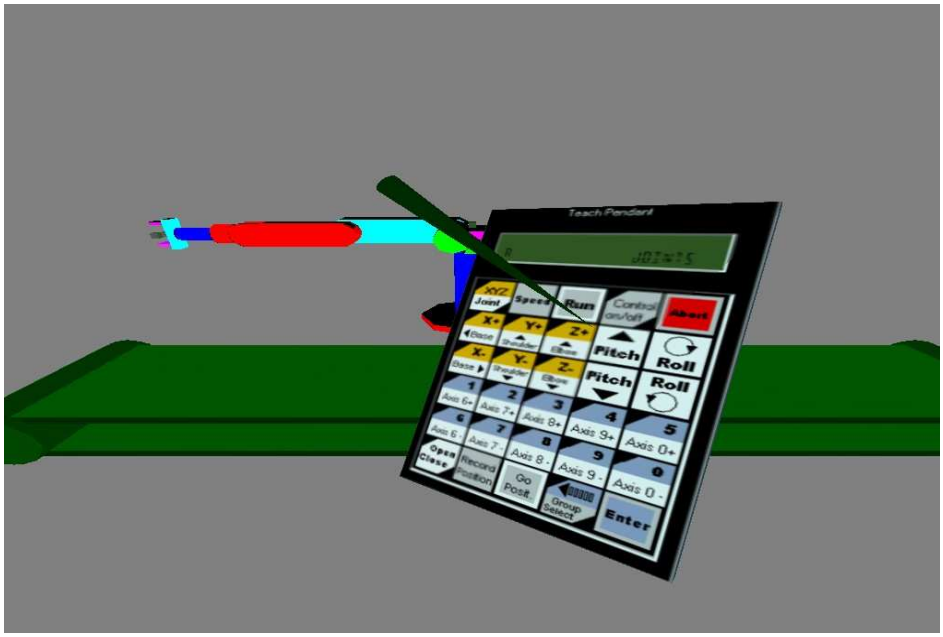


Figura 6.17 - Caneta sobre o TP virtual

Para detectar se a caneta tocou uma certa tecla do TP, tanto as teclas, quanto a caneta foram modeladas como objetos geométricos entre os quais se procura avaliar se existe ou não uma colisão. Os detalhes da implementação do controle de colisão são apresentados na seção 6.7.

O corpo do TP virtual foi modelado por um paralelepípedo com as mesmas dimensões do TP real, ou seja, 2.5 cm de largura por 8 cm de altura e 0.25 cm de profundidade.

Visualmente as teclas do TP virtual foram modeladas com a técnica do mapeamento de texturas. Essa técnica consiste em “colar” uma imagem qualquer sobre as faces em um objeto. No

caso foi colada uma imagem do TP real na face frontal do paralelepípedo simulando as teclas reais conforme a Figura 6.18.

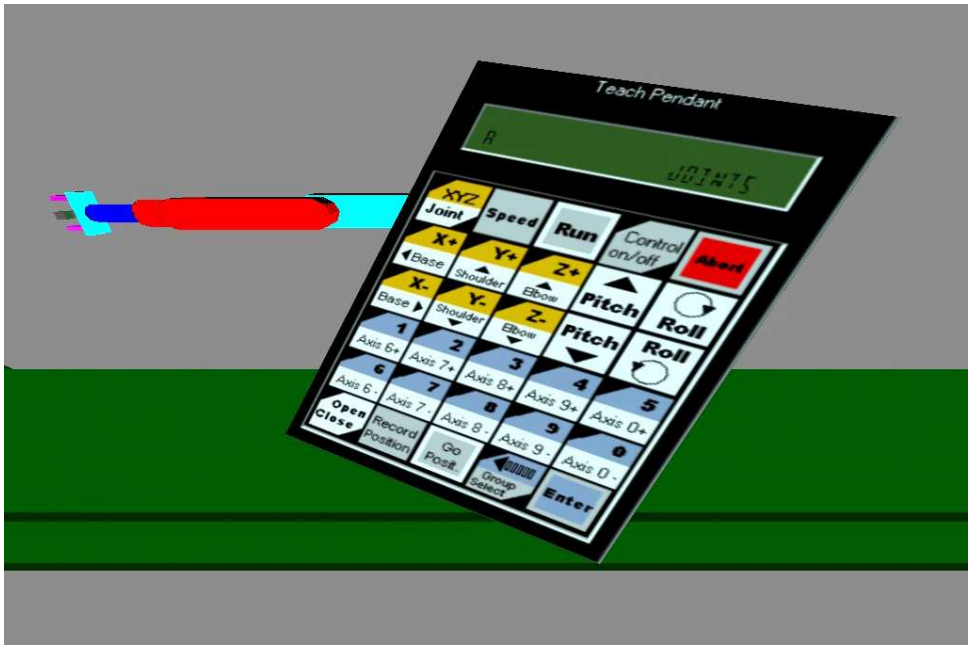
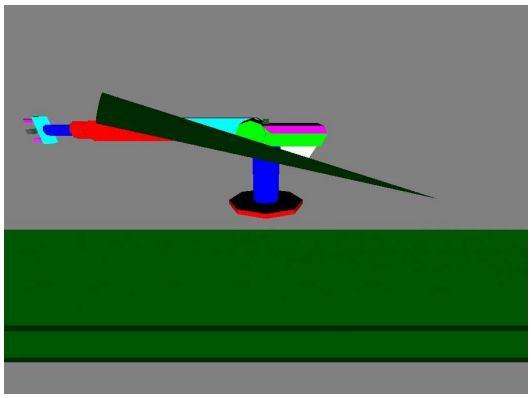


Figura 6.18 – TP com a textura que representa os botões

No caso do apontador virtual, utilizou-se, para sua modelagem, de uma função da biblioteca GLUT chamada `glutSolidCone`, que desenha um cone, conforme mostra a Figura 6.19.



```
void DrawCone(void)
{
    glutSolidCube(0.2);
}
```

Figura 6.19 – Representação geométrica do apontador

6.6 Controle de colisão do Teach Pendant

Para o tratamento de colisão entre a apontador e o TP é utilizada a técnica dos “envelopes”. Os envelopes são representados por cubos que envolvem os objetos e que servem para simplificar a geometria destes, tornando o teste de colisão mais rápido. A SmallVR [PIN 02] fornece uma classe para esse fim.

Para cada tecla existente no TP foi modelado um envelope e posicionado exatamente sobre cada tecla, conforme a Figura 6.20.

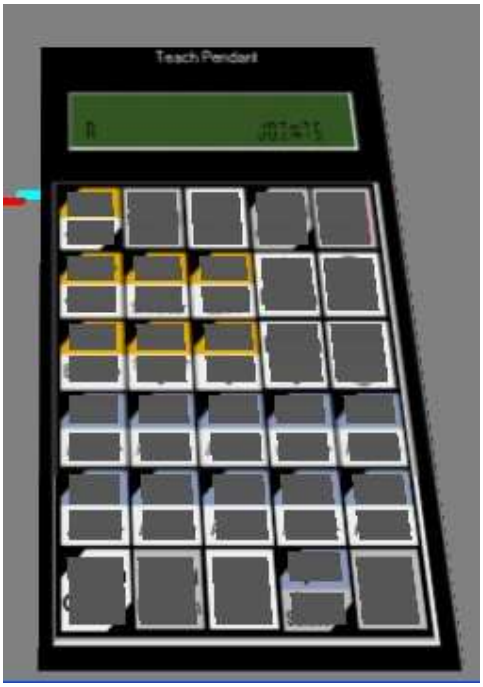


Figura 6.20 – Envelopes dos botões

Ao apontador, por sua vez, também foi associado um envelope. Esse envelope é implementado da mesma forma como foi feito nos botões do TP, com a diferença de que o envelope envolve somente na ponta do cone que representa o apontador (Figura 6.21), significando que somente será detectada a colisão quando a ponta do cone colidir com um dos botões do TP.

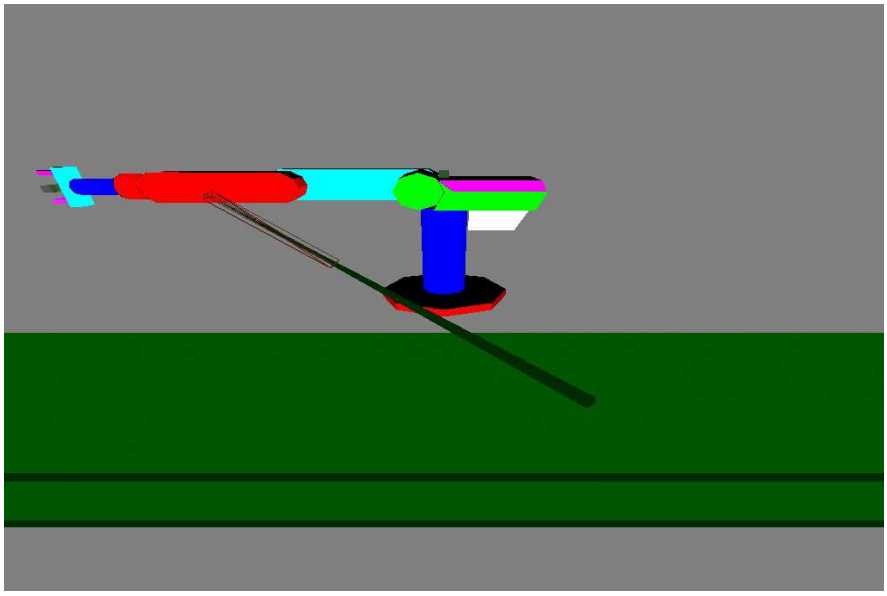


Figura 6.21 – Envelope da caneta

Na SmallVR estes objetos envelopes são representados pela classe SmVR_CEnvelope que estende a classe `SmVR_CGeometricObject`, instanciando sempre um cubo de 12 faces triangulares, e possui um método (colided) para detectar colisão com outro envelope. Tendo-se os dois envelopes, o teste de colisão é efetivamente feito através do método **int** `SmVR_CEnvelope::colided(SmVR_CEnvelope *obj)` que devolverá 1(um) se houver colisão e 0 (zero) se não houver. O trecho de código abaixo (Figura 6.22) exemplifica o teste de colisão entre os envelopes `botãoEnvelope` e `apontadorEnvelope`. Isto torna a detecção muito simples, bastando para tanto testar a intersecção das áreas dos envelopes.

```
if(botaoEnvelope->colided(apontadorEnvelope))
      printf("colidiu %d\n",++cont);
```

Figura 6.22 – Chamada para função de teste de colisão

6.7 Interação do Teach Pendant com o usuário

No ambiente virtual é necessário possibilitar ao usuário a interação com os objetos *teach pendant* e *apontador*. Para isso foram utilizados rastreadores de posição *Polhemus Isotrack II* que capturam o movimento de dois pontos no espaço.

Na SmallVR, a utilização de um dispositivo de realidade virtual é feita através da classe `SmVR_CDevice`.

Nos objetos virtuais do teach pendant e do apontador existem objetos virtuais que estão atrelados aos rastreadores reais. Isto possibilita que o TP e o apontador sejam movidos no ambiente virtual de acordo com as coordenadas do Isotrack. No *Teach Pendant* e no *Apontador* há objetos que representam a posição dos rastreadores como nos objetos reais. Isto pode ser visto na Figura 6.23 e Figura 6.24.

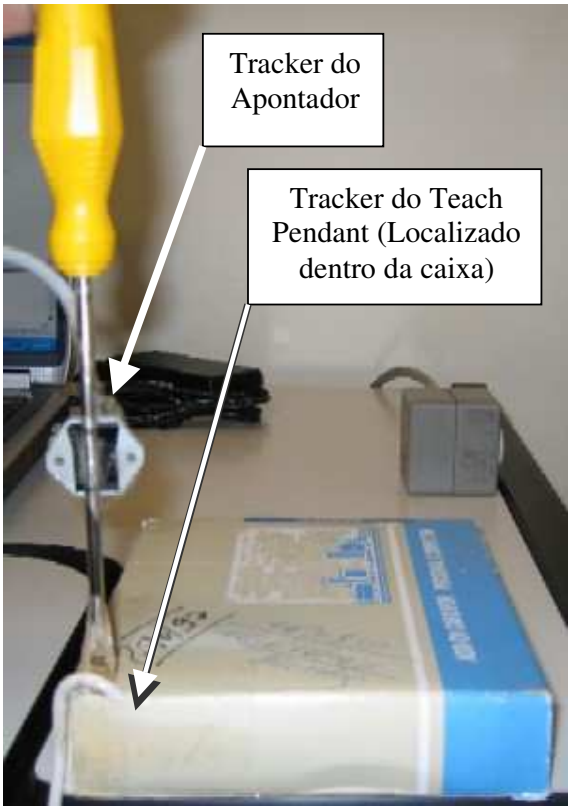


Figura 6.23 – Rastreadores de posição do TP e da caneta

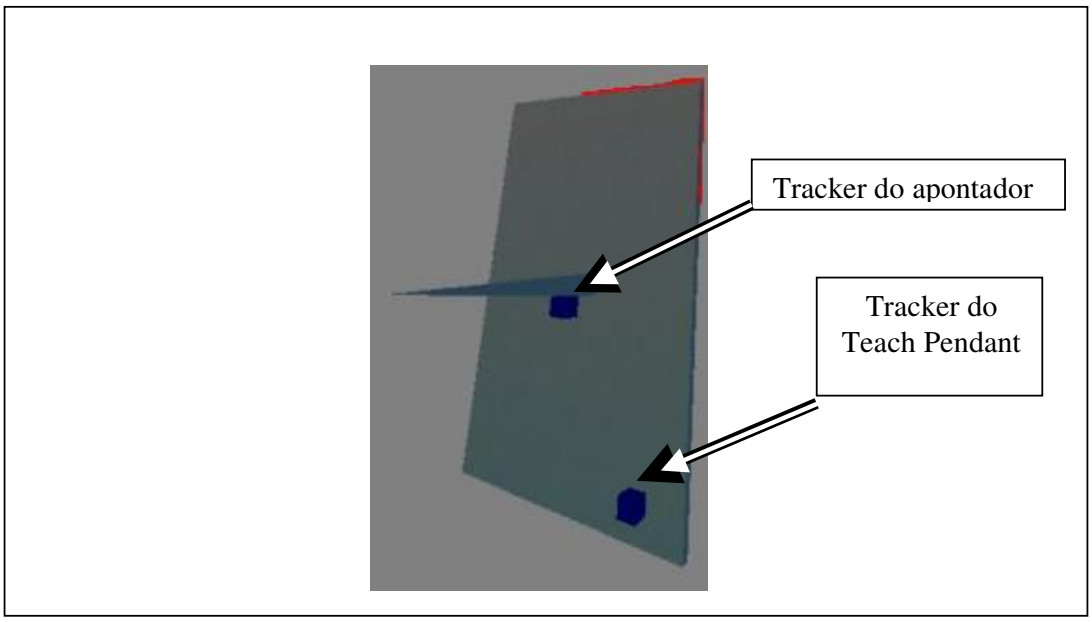


Figura 6.24 – Objetos representantes dos rastreadores do TP e da caneta

Tanto o TP quanto o apontador foram estruturados virtualmente em uma hierarquia que possibilita mover os objetos pelo ambiente através da movimentação do rastreador. Essa hierarquia é composta por objetos vazios (objetos sem forma geométrica) que se encontram exatamente na posição (do TP ou da caneta) que o rastreador real se encontra em relação ao objeto real. O primeiro objeto vazio, que se encontra ligado ao TP e ao *apontador*, é o *TrackerObject* e o *TrackerObject2* respectivamente e estes estão diretamente ligados aos seus respectivos rastreadores reais. Acima do *TrackerObject e TrackerObject2*, temos ainda o objetos chamados *PaiObject e PaiObject2 respectivamente*, que servem basicamente para aplicação de transformações geométricas. Esses objetos são necessários, pois não é possível aplicar as transformações geométricas diretamente nos objetos que representam os rastreadores (*TrackerObject e TrackerObject2*). Isso porque a cada movimento dos rastreadores todas as transformações geométricas dos trackersObjects são "zeradas". Os rastreadores virtuais e suas hierarquias estão demonstrados na Figura 6.25 (TP) e Figura 6.26 (*Apontador*). No final temos um esquema montado dessa forma: rootObject -> PaiObject -> TrackerObject -> teachPendant e rootObject -> PaiObject2 -> TrackerObject2 -> Apontador.

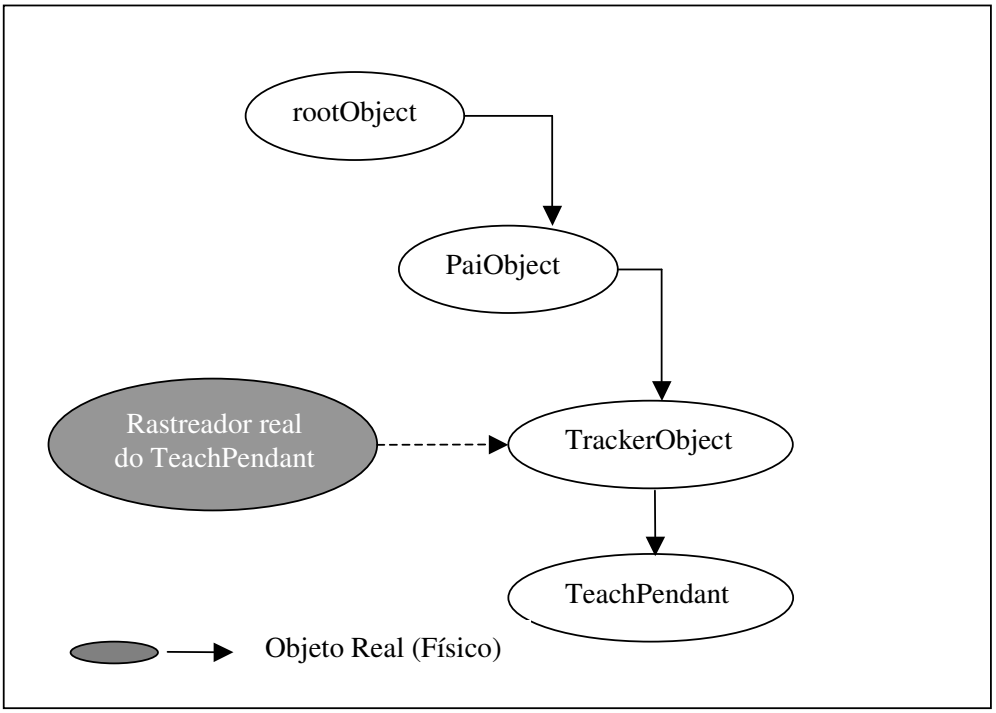


Figura 6.25 – Hierarquia de objetos do TP

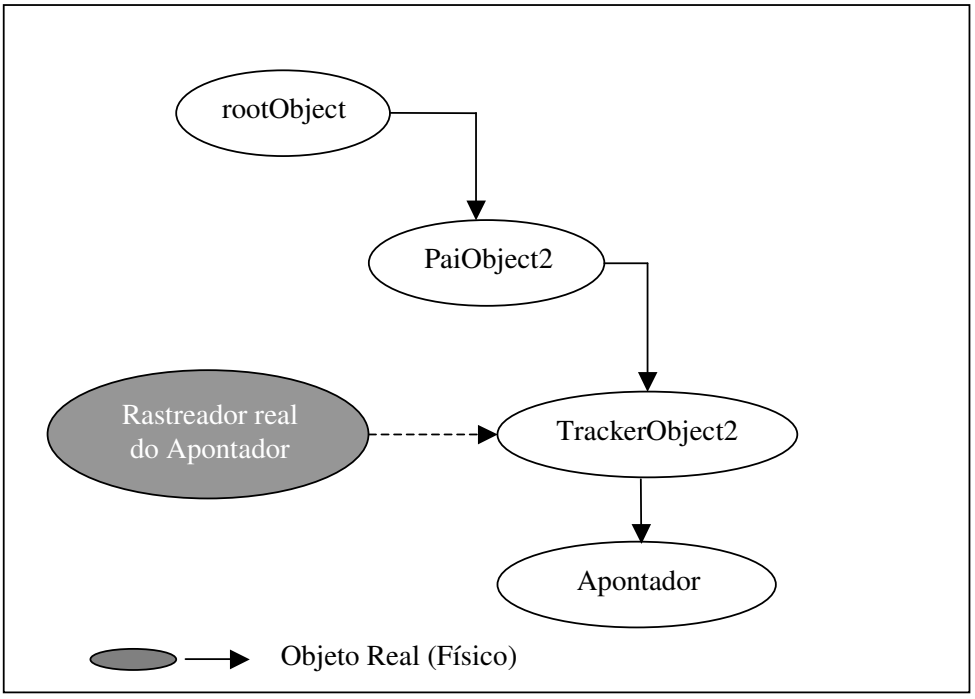


Figura 6.26 – Hierarquia de objetos do apontador

6.8 Navegação na célula robotizada

A navegação é um dos pontos que tem grande importância para que o sistema tem uma melhor usabilidade que o ASIMOV, pois pretende-se que o usuário deve se habite facilmente com o modo como se locomove na célula robotizada, com isto consegue atingir os pontos desejados de maneira pratica.

Para atingir este objetivo foi adotado um método de navegação em que a direção do movimento é dada pela direção para onde aponta a testa do usuário e o sentido é definido por dois botões (para frente e para trás) localizados ao lado do Teach Pendant como pode ser visto na Figura 6.27.

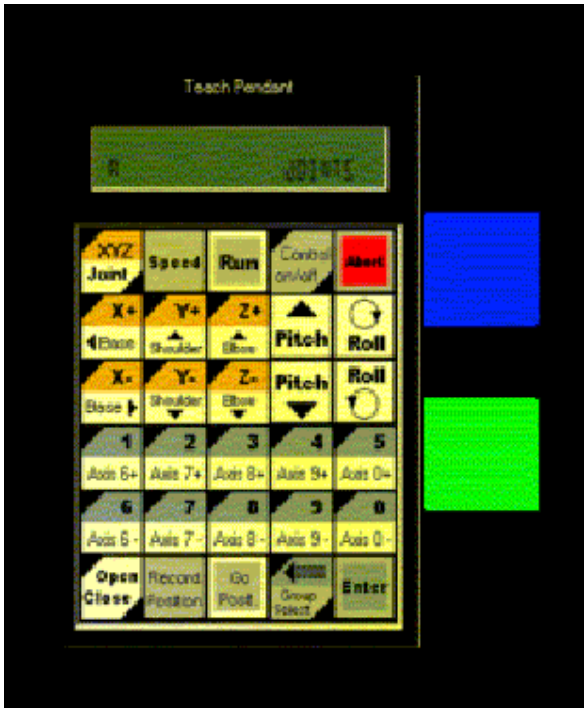


Figura 6.27 – Botões de navegação

Inicialmente para representar o observador e o alvo define-se uma hierarquia em que um objeto "*Usuário"* é colocado como pai do objeto "*Alvo"*. A seguir liga-se a este *Usuário* um rastreador de posição, como filho do objeto *Usuário* coloca-se o objeto alvo.

Para permitir translações com o conjunto observador-alvo, deve-se criar um objeto vazio como pai do *Usuário.* Na Figura 6.28 temos representada como fica a hierarquia recém mencionada.

Para que o ambiente se mova conforme o movimento do *Usuário* é necessário que na chamada `gluLookAt(),` se tenha acertado os parâmetros para a posição do *Usuário* e do *Alvo,* como mostra a Figura 6.29.

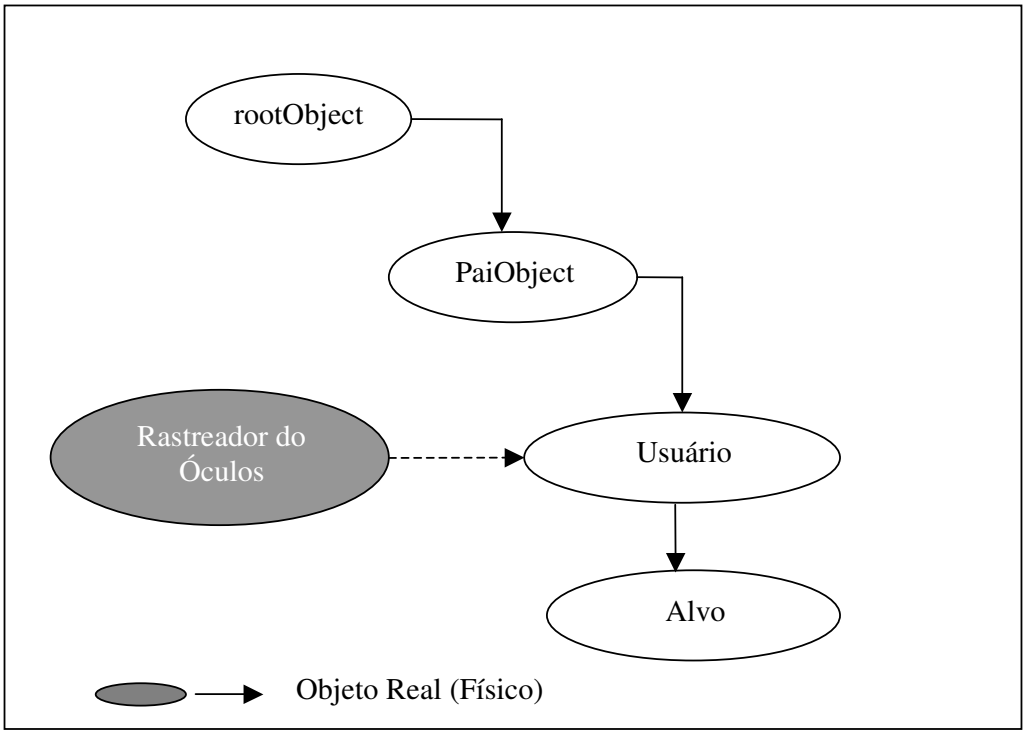


Figura 6.28 – Hierarquia navegação

```
//pt_user é Usuário
//pt_target é Alvo
gluLookAt(pt_user.X,pt_user.Z, pt_user.Y,
          pt_target.X,pt_target.Z,-pt_target.Y,
          0.0f,1.0f,0.0f);
```

Figura 6.29 – Chamada da função gluLookAt

Feitos os passos descritos cima apenas é possível obter uma visão da célula robotizada, entretanto ainda não é possível movimentar-se sobre ela além dos limites de alcance do rastreador. Para que isto seja feito foram criadas os botões mostrados na Figura 6.27, que têm por objetivo

fazer a movimentação. Quando é pressionado o botão superior é feita uma translação do objeto *Usuário* para frente, todavia esta transformação deve ser feita no sistema de coordenadas do *Alvo* para que o movimento aconteça na direção em que o usuário estiver olhando. O botão inferior tem a funcionalidade oposta, ou seja, move o *Usuário* para trás.

7. Avaliação da Eficácia do Sistema Imersivo

Havia a pretensão de se fazer avaliações sobre o novo sistema com o objetivo de verificar a sua eficiência comparativamente com o sistema ASIMOV, entretanto não houve tempo hábil para que isto fosse feito. Então serão propostos neste capítulo alguns testes que se julga pertinentes para a avaliação do sistema.

Esses testes devem feitos de em duas etapas. A primeira etapa é feita através da coleta automática apenas com o usuário usando os sistema imersivo e o ASIMOV. A segunda etapa é apresentar um questionário sobre a opinião de cada usuário sobre os sistemas.

Na primeira etapa da avaliação tem como objetivo passar pontos pré-determinados no espaço representados como objetos no ambiente, os quais o usuário deve alcançar e pegar. O usuário deve movimentar o robô no sistema ASIMOV e no sistema imersivo. Com as observações julga-se possível verificar o numero de erros cometidos e o tempo gasto para que os pontos sejam atingidos pela garra do robô. Com isto é possível verificar a eficiência do uso da perspectiva dos ambientes e, se a ferramenta de movimentação no ambiente imersivo produz uma melhora que justifique sua utilização.

Poderá ser avaliado ainda a sensação descrita pelos usuários no uso do óculos 3D e do *teach pendant* virtual. Os itens a serem avaliados são conforto e facilidade de manipulação do sistema. O item conforto refere-se ao fato do usuário se sentir bem usando o sistema. A manipulação poderá ser abordada nessa parte de uma forma mais subjetiva onde se buscará a opinião do usuário sobre qual o melhor sistema para manipulação do robô. Um esboço do questionário pode na Figura 7.1:

Figura 7.1 Questionário pós-teste

1 – Como você se sentiu usando o sistema em relação ao equipamento?

2 – Qual sua opinião em relação ao uso do novo sistema?

3 – Em relação à manipulação do ambiente você acredita que o sistema virtual é mais fácil de usar?

Os testes permitirão se avalie as vantagens e desvantagens do sistema imersivo em relação ao sistema ASIMOV. Além disto acredita-se ser possível a obtenção de dados que mostrarão a eficácia do sistema imersivo no que diz respeito à exatidão da captura dos pontos quando comparado ao sistema atual. Do ponto de vista do usuário será possível avaliar a adaptação e aceitação dos usuários quanto ao ambiente imersivo e suas interfaces.

8. Considerações Finais

No novo ambiente imersivo foi possível a inserção de várias técnicas que mostram-se úteis para a facilidade de aprendizagem do usuário em sistemas, principalmente de treinamento. Temos entre elas implementadas no sistema a técnica de mapeamento de objetos reais no ambiente virtual através de rastreadores de posição, como é o caso do modo de manipulação do Teach Pendant, também a utilização do óculos 3D que proporciona uma imersão no sistema aproximando a utilização ao uso em ambientes reais.

Foi possível fazer um interação entre o programa já existente com o programa imersivo, tendo assim uma utilização dos ambientes já criados e um reutilização de trabalhos já feitos. E também a geração de relatório de pontos em que o usuário pode testar os resultados em outros sistemas e inclusive fazer relatórios dos mesmos.

Por escassez de tempo houve alguns quesitos que não puderam ser agregados ao sistema como por exemplo estereoscopia, que traria um realismo maior ao ambiente e também não foi possível fazer a inserção de mensagens no visor do Teach Pendant, as mensagens são mostradas em um canto da tela. Os testes não puderam ser realizados para realmente comprovar a eficácia do novo ambiente, entretanto foram propostos testes que se julga pertinentes para esta avaliação.

Como tópicos a serem explorados em trabalhos futuros ressaltam-se a implementação de estereoscopia, uma maior integração final com o ambiente ASIMOV além de um teste da aceitação dos usuários frente a essa nova abordagem de interação com este ambiente.

9. Referências

[ASI 99] ASIMOV – Simulador de Robótica – Células de Produção, Manual do Usuário, SENAI-RS, Porto Alegre 1999;

[BIL 97] Billinghurst, M., Baldis, S., Matheson, L., Phillips, M., 3D pallette, a virtual reality content creation tool. Proceedings of VRST'97. 1997. ACM. pp. 155-156;

[BOW 98] Bowman, D., Wineman, J., Hodges, L., Allison, D., Designing Animal Habitats Within an Immersive VE . IEEE Computer Graphics & Applications, 1998. 18(5): pp. 9-13;

[CVC 03] GRAPHICS, Parallel, Cortona VRML Client, disponível (12/11/03) em: http://www.parallelgraphics.com/products/cortona/;

[DEL 02] DELMIA SYSTEMS, The simulation and analysis tool for machine tools and machining operations. Disponível *(15/08/03)* em: http://www.delmia.com/ ;

[EXH 97] EXHIBITORS, *Virtual reality in manufacturing research and education*;

[GAR 02] GARCIA, Fabiano Luiz Santos, CAMARGO, Fábio Doneda, LORENZATO Lucas, Virtual TrainingPit Um Sistema de Treinamento Virtual de Pilotos de Aeronaves, Programa de Pós-Graduação da Eng. de Produção, UFSC. Disponível (01/09/03) em: http://www.lrv.eps.ufsc.br/recursos/artigos/VRtrainingpit.pdf ;

[HOL 96] HOLERBACH, John M., University of Utah Department of Computer Science 3190 Merrill Engineering Building Salt Lake City, UT 84112. Charles W. Wampler Mathematics Department General Motors Research and Development Box 9055 Warren, MI 48090 disponível em (15/06/04): http://www.cs.utah.edu/~jmh/Papers/ijrr96.ps ;

[HOS 96] HOSSEINI, Mojtaba, GEQRGANAS Nicolas D., Collaborative Virtual Environments for Training, Multimedia Communication Research Laboratory School of Information Technology and Engineering University of Ottawa Canada. Também disponível (01/09/03) em: http://www1.acm.org/sigs/sigmm/MM2001/ep/hosseini-demo/index.html ;

[INT 95] INTELLIGENT MANUFACTURING *(1995). Virtual Reality is for real,* Volume. 1, Número 12.

[LIN 99] Lindeman, R., Sibert, J., Hahn, J., Hand-held Windows: Towards Effective 2D Interaction in Immersive Virtual Environments. Proceedings of VR'99. 1999. IEEE. pp.205-212;

[LOF 99] LOFTIN, R. Bowen, PETTITT, B. Montgomery, SU, Simon, CHUTER, Chris, MCCAMMON, J. Andrew, DEDE, Chris, BANNON, Brenda, ASH, Katy, **PaulingWorld: An** Immersive Environment for Collaborative Exploration of Molecular Structures and Interactions, disponível (01/09/03) em: http://www.vmasc.odu.edu/vetl/html/ScienceSpace/PW.pdf ;

[MAC 03] MACHADO, Liliane dos Santos, Pesquisa e Desenvolvimento de Sistemas de Realidade Virtual para Treinamento em Oncologia Pediátrica, Projeto FAPESP #99/01583-0, disponível (01/09/03) em: http://www.lilianesm.hpg.ig.com.br/rvmed_p.html ;

[NET 01] NETTO, Antonio Valerio, OLIVEIRA, Maria Cristina Ferreira, Realidade Virtual Empregada na prototipação de Equipamentos, Instituto de Ciências Matemática e de computação - ICMC/USP, Também disponível (01/09/03) em: http://www.propg.ufscar.br/publica/4jc/posgrad/resumos/0044-valerio.htm ;

[PIN 98] PINHO, Márcio Serolli, DIAS, Leandro Luis, MOREIRA, Carlos, BECKER, Gustavo, KODJAGHLANIAN, Emmanuel, DUARTE, Lúcio Mauro, Criação de um Dispositivo de Navegação em Mundos Virtuais (Bicicleta Virtual), Disponível (01/09/03) em: http://grv.inf.pucrs.br/Pagina/Projetos/Bike/Bike.htm ;

[PIN 02] PINHO, Márcio Serolli, SmallVR A simple toolkit for VR application development, Virtual Reality Group at Pontifícia Universidade Católica do Rio Grande do Sul, disponível (06/11/03) em: http://grv.inf.pucrs.br/Pagina/SmallVR/SmallVR.html;

[SAB 99] SABBATINI, Renato M.E., Realidade Virtual no Ensino Médico, Revista Informática Médica Volume 2 numero 2 Mar/Abr 1999, disponível (01/09/03) em: http://www.epub.org.br/informaticamedica/n0202/sabbatini.htm ;

[SIL 02] SILVA, Cledja Karina Rolim da, Realidade Virtual na Educação, Disponível (01/09/03) em: http://www.cin.ufpe.br/~if124/apresentacoes/2002_09_03_ckrs/rv_educacao.ppt ;

[SUN 03] Sun Microsystems What's Happening about Ivan E. Sutherland, disponível (01/09/03) em: http://www.sun.com/960710/feature3/alice.html#pics ;

[WVF 04] Maya Wavefront OBJ - disponível (09/06/04) em: http://www.alias.com/eng/products-services/maya/new/index.shtml.

[VRM 95] BELL, Gavin, PARISI, Antonhy, PESCE, Mark, VRML - The Virtual Reality Modeling Language – Version 1.0 Specification, disponível (12/11/03) em: http://www.web3d.org/technicalinfo/specifications/VRML1.0/index.html.