ESCOLA POLITÉCNICA
Universidade de São Paulo

Desenvolvimento e estudo de um Desktop3D com o uso de gestos

Grupo de Trabalho:
Alexandre Picelli Vicentim
Daniel Makoto Tokunaga
Marcos Massaharu Muto

São Paulo
2007

Alexandre Picelli Vicentim
Daniel Makoto Tokunaga
Marcos Massaharu Muto

Desenvolvimento e estudo de um Desktop3D com o uso de gestos

São Paulo
2007

Alexandre Picelli Vicentim
Daniel Makoto Tokunaga
Marcos Massaharu Muto

Desenvolvimento e estudo de um Desktop3D com o uso de gestos

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro de Computação

Área de concentração:
Engenharia de Computação

Orientador: Profº Dr. Romero Tori
Co-orientador: João Luiz Bernardes Jr.

São Paulo
2007

DEDICATÓRIA

Dedico este trabalho aos meus pais, Carlos Alberto Vicentim e Maria Marcina Picelli Vicentim, pela colaboração e apoio oferecidos ao longo de minha formação.

Alexandre Picelli Vicentim

Dedico este trabalho a minha família, a todos que apoiaram a minha formação e para o futuro dos amigos integrantes do projeto.

Daniel Makoto Tokunaga

Dedico este trabalho a minha mãe Luzia Mitiyo Sato Muto e meu irmão Marcio Massayoshi Muto pelo apoio e suporte. E em memória de José Kenji Muto.

Marcos Massaharu Muto

AGRADECIMENTOS

Ao professor Romero Tori, pela orientação e pelo constante estimulo transmitido durante todo o trabalho e ao João Luiz Bernardes Junior, pela orientação técnica e por todo apoio realizado.

À professora Lucia Filgueiras pelo apoio dos testes de usabilidade, aos amigos do InterLab, pela ajuda oferecida durante o projeto, ao amigo Fernando Picelli Vicentim pela participação nos testes.

E a todos que colaboraram direta e indiretamente, na execução deste trabalho.

RESUMO

Interfaces naturais e 3D estão em uma crescente demanda e apresentam desafios interessantes e oportunidades de pesquisa. Este projeto apresenta a pesquisa, desenvolvimento e teste de um ambiente de organização de arquivos virtual que permite empilhamento de objetos e interação através dos movimentos da mão do usuário, sem a necessidade de qualquer dispositivo, exceto por uma câmera. HandVu é usado para rastreamento da mão e reconhecimento de postura, integrado à um engine didático de jogos que manipula o ambiente virtual. Essa integração é descrita e pode ser vista como um segundo objetivo deste trabalho, permitindo futuros desenvolvimentos de diversos ambientes virtuais, jogos ou aplicações com interfaces baseadas em gestos. A medição de desempenho e usabilidade e respostas subjetivas dos testes com inúmeras pessoas mostraram que o sistema resultante é altamente dependente do hardware mas também mostraram que interfaces naturais e 3D com interfaces baseadas em gestos possuem um grande potencial.

Palavras-chave: Interfaces 3D, Desktop 3D, Interfaces baseadas em gestos, visão computacional, engine de jogos.

ABSTRACT

Natural and 3D interfaces are in increasing demand and present interesting challenges and opportunities for research. This project presents the research, development and testing of a virtual desktop environment that allows object stacking and interaction through the movements of the user's own hand, without any devices, except for a webcam. HandVu is used for hand tracking and posture recognition and integrated with a didactic game engine that handles the virtual environment. This integration is described and can even be seen as a secondary objective of this work, allowing future implementations of many diverse virtual environments, games or applications with gesture-based interfaces. Objective performance and usability measures as well as subjective responses from tests with several users show that the resulting system is heavily dependent on hardware but also shows that gesture-based 3D and natural interfaces have great potential.

Keywords: 3D interfaces, 3D Desktop, Gesture-based interfaces, computer vision, game engine.

内容梗概

現在、直感的３D インターフェイスの需要は増えつつあり、そして研究課題としては興味深いチャレンジとチャンスが存在する分野である。本プロジェクトでは物体の積み重ねが可能であり、ウェブカメラ以外の特殊な機材を必要とせず、ユーザー本人の手の動作みでインタラクション可能なヴァーチャルデスクトップの研究、製作、テストを行った。ユーザーの手の追跡と手指の形状の区別には HandVu が使用され、仮想環境を構築、コントロールする教材用のゲームエンジンとの統合がなされた。この統合は本プロジェクトで記録、紹介されており、二次的目的としても見て取れ、将来様々なジェスチャーをインターフェイスとして使用した仮想環境や、ゲーム、アプリケーションなどの製作を可能とする。本プロジェクトではパフォーマンステストやユーザビリティーテストの客観的な測定や様々なユーザーのテストによる主観的な意見の採取の結果、システムがハードウェアの性能に大きく依存して事、しかしジェスチャーを使用した直感的３D インターフェイスが大きな可能性を持っている事が窺える。

Keywords: ３D インターフェイス、３D デスクトップ、ジェスチャーベースインターフェイス 、コンピューターヴィジョン、ゲームエンジン。

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS DE SIGLAS

DLL	Dynamic-lynk library
FPS	Frames per second
JNI	Java Native Interface
UML	Unified Modeling Language
GPU	Graphics Processor Unit
JRE	Java Runtime Environment

SUMÁRIO

1. INTRODUÇÃO

Atualmente há uma crescente demanda por diferentes formas de interação com sistemas computacionais, como pode ser visto através da grande aceitação do público a produtos que não utilizam apenas meios convencionais de interação, como por exemplo, o iPhone da Apple e o Surface da Microsoft.

Outra crescente vertente é a utilização de ambientes em três dimensões, como pode-se observar em jogos, aplicações de realidade virtual, CAD (Computer-Aided Design), entre outros. Contudo, as interfaces de interatividades convencionais (teclado e mouse) nesses ambientes não se mostraram tão eficientes quanto em ambientes em duas dimensões, visto que esses dispositivos são utilizados em duas dimensões (BOWMAN et al., 2003).

Tentando melhorar a interatividade em ambientes tridimensionais, novos dispositivos são criados, como o console de videogame Wii da Nintendo (BERNARDES, 2007), que faz o uso de gestos com um bastão para o controle dos jogos. Desse modo, as ações dos personagens são baseadas no movimento do jogador, ou seja, em um jogo de tênis a pessoa precisa realizar os movimentos como se estivesse em uma partida real.

Nesse contexto, observa-se o potencial do uso de gestos como forma de interação, uma vez que torna a interface mais intuitiva, diminui-se a curva de aprendizado da aplicação e mantêm o foco do usuário em suas tarefas, de modo que o mesmo não precisa parar suas ações para pensar em como executá-las (PAVLOVIC; SHARMA; HUANG, 1997).

Assim, esse trabalho tem como objetivo desenvolver um sistema que consiste em um ambiente de trabalho de organização de arquivos e programas (um "desktop") manipulado através de gestos manuais e com uma interface em três dimensões, de forma que o modo de interação e a interface se aproximem de um ambiente real, sendo possível mover livremente os arquivos dentro do ambiente, inclusive para empilhá-los. É também objetivo do trabalho a verificação de quão bom é a utilização dos gestos como meio de interação nesse ambiente.

Para isso, o ambiente será uma sala (o interior de um cubo) que conterá blocos que representam os arquivos e programas. Toda a manipulação dos blocos será realizada através dos movimentos da mão do usuário e as posturas da mão determinarão as ações realizadas dentro do ambiente.

Os blocos serão identificados na face superior com uma imagem representando o programa que abre o arquivo (essa imagem será a mesma que é utilizada nos ícones do sistema operacional Windows) e nas laterais pelo nome do arquivo, de forma semelhante a livros no ambiente físico. Os arquivos podem ser movidos dentro do ambiente de trabalho para qualquer lugar do espaço, sendo possível mover para cima de outros arquivos criando uma pilha, o que deve facilitar a organização dos mesmos.

Também é possível mover as pilhas como os arquivos e há um modo para visualização do conteúdo da pilha, sendo possível selecionar parte da mesma.

Para facilitar a visualização das pilhas e blocos, há uma câmera dentro do ambiente que pode ser rotacionada e ser movimentada no eixo vertical e que, além disso, tem um mecanismo de zoom.

1.1 OBJETIVO

O objetivo deste projeto é desenvolver um ambiente de trabalho (desktop) em três dimensões e com interface por gestos e também estudar esse ambiente do ponto de vista de usabilidade, determinando as vantagens, desvantagens e possíveis melhorias para essa forma de interação. Além de ser motivado pela aceitação do público por essas novas interfaces, como mencionado anteriormente, o projeto apresenta desafios tecnológicos que o tornam bastante interessante como projeto de formatura. Para o reconhecimento de gestos, é utilizado o aplicativo HandVu, um software desenvolvido por Mathias Kolsch em sua tese de doutorado (KOLSCH, 2004). O HandVu foi escolhido por ser, até o momento, a API mais conhecida, gratuita e

de código aberto que rastreia o movimento da mão e reconhece posturas, determinadas no aplicativo, em tempo real (KOLSCH, 2004).

O HandVu terá sua funcionalidade estendida para que a profundidade da mão com relação a câmera também seja detectada e para que as funcionalidades necessárias ao projeto possam ser acessadas de aplicações Java, visto que essa será a linguagem adotada para o desenvolvimento, juntamente com a ferramenta enJine (ENJINE, 2007). A extensão das funcionalidades do HandVu e sua integração com o enJine, além de serem passos necessários para o desenvolvimento do projeto, podem ser considerados isoladamente como um importante objetivo secundário do projeto, criando um framework para facilitar o desenvolvimento de futuras aplicações com interfaces baseadas em gestos utilizando o enJine. Inicialmente não faz parte dos objetivos a execução/visualização dos arquivos presentes no desktop, funcionalidade que só será implementada caso haja tempo ao fim do projeto.

1.2 MOTIVAÇÃO

Atualmente pode-se notar uma vertente dos aplicativos eletrônicos que visa uma maior interatividade e usabilidade com relação ao usuário, como por exemplo:

iPod: Possui uma interface simples e organizada, de modo que as músicas são facilmente encontradas. Além disso, o iPod possui uma interface de interação com o usuário intuitiva, utilizando um "pad" sensível ao toque, sendo necessário apenas deslizar o dedo sobre o "pad" para acessar o conteúdo do aplicativo. (IPOD, 2007)

Wii: O console de videogame Wii, utiliza um bastão que captura os movimentos do jogador. Desse modo, os jogos tornam-se mais intuitivos, pois não necessita de comandos complicados, como a necessidade de apertar uma seqüência de botões para movimentação do personagem. (WII, 2007)

iPhone: O aparelho de celular da Apple, além de receber e fazer ligações, possui outras funcionalidades como tocar músicas e visualizar fotos. O iPhone

possui tela sensível a toque, o que aumenta a interatividade com o usuário. (IPHONE, 2007)

EyeToy: Uma câmera digital a cores, semelhante a uma webcam, para o PlayStation2. A tecnologia faz uso da visão computacional para processar imagens tiradas pela câmera, o que permite a interação com jogo através do movimento do corpo, detecção de cores e som, através do microfone incorporado. (EYETOY, 2007)

Com todo o sucesso desses aplicativos no mercado, pode-se concluir que há uma tendência dos aplicativos serem mais intuitivos e interativos. Esse projeto, então, visa explorar essa tendência através de um sistema que utiliza gestos para interagir com sua interface, uma vez que o uso de gestos pode tornar a interação com o sistema mais intuitiva, possibilitando ao usuário utilizar os movimentos habituais para realizar determinada tarefa em seu cotidiano (BOWMAN et al., 2003). Assim, o tempo de aprendizado e a dificuldade para utilizar aplicativos que utilizam essa técnica de interação podem ser diminuídos. As vantagens e eventuais desvantagens dessa forma de interação, da forma como está presente no HandVu, é justamente um dos pontos que se deseja estudar nesse projeto.

1.3 ORGANIZAÇÃO

O trabalho está dividido nos seguintes capítulos: Introdução, Aspectos Conceituais, Especificação, Metodologia, Projeto e Implementação e Testes.

Na introdução é dada uma visão geral do projeto, com seus objetivos, motivação e a estrutura da monografia.

Em aspectos conceituais são apresentadas as tecnologias estudadas para realização do projeto, assim como os conceitos teóricos envolvidos.

Na especificação são apresentados os requisitos, e alguns diagramas utilizados para especificar o sistema.

No capítulo de metodologia é apresentada a metodologia utilizada para desenvolvimento desse projeto, assim como os motivos que levaram a sua escolha.

Em projeto e implementação são detalhados os passos para a implementação de todas as etapas do projeto.

Em testes são apresentadas todas as metodologias que foram utilizadas para realização de testes.

No capítulo de considerações finais são apresentados os resultados dos testes que foram especificados no capítulo de testes, assim como as conclusões a respeito da utilização dos gestos e de ambientes em três dimensões.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo é dividido em duas partes primeiro são apresentados os conceitos utilizados, depois se descreve as tecnologias utilizadas, para, assim, facilitar a compreensão das ferramentas abordadas.

2.1 REVISÃO CONCEITUAL

São apresentados os termos e conceitos utilizados no trabalho de modo a apresentar o contexto do sistema desenvolvido.

2.1.1 REALIDADE AUMENTADA

Realidade aumentada é a sobreposição de objetos virtuais tridimensionais com um ambiente real por meio de algum dispositivo tecnológico, feita em tempo real e com registro em 3D entre os objetos reais e virtuais (AZUMA, 1997). Porem esse conceito fica melhor explicado quando inserido dentro da Realidade Misturada, que representa a mistura do real com o virtual e assim gerando duas novas possibilidades: Realidade Aumentada, onde o ambiente predominante é o mundo real, e Virtualidade Aumentada, na qual o ambiente predominante é o mundo virtual. Dessa forma podemos perceber que a realidade aumentada é uma particularização da realidade misturada (MILGRAM et al., 1994).

Neste projeto, utilizou-se inicialmente a Virtualidade Aumentada, visto que a mão do usuário era incluída dentro do ambiente virtual do Desktop 3D. Essa inserção da mão do usuário, no entanto, foi analisada através da criação de um protótipo. Esse protótipo foi comparado com outro no qual uma mão virtual interage com o desktop. Os resultados dessa comparação levaram a

substituição da Virtualidade Aumentada (utilizando a imagem da mão real) pela solução de Realidade Virtual (com a mão virtual).

Uma possível extensão do projeto é a projeção do Desktop3D em ambiente real apropriado, para permitir que o usuário interaja com sua própria mão com o ambiente virtual do desktop, de forma semelhante à Microsoft Surface (SURFACE, 2007).

2.1.2 REALIDADE VIRTUAL

Realidade Virtual é uma interface avançada para aplicações computacionais, onde o usuário pode navegar e interagir, em tempo real, em um ambiente tridimensional gerado por computador, usando dispositivos multisensoriais (KIRNER; PINHO, 1997).

2.1.3 GESTOS

A cada dia os computadores têm se integrado cada vez mais à sociedade, tornando a interação humano-computador (IHC) ainda mais importante. Nesse contexto, o uso de gestos provê uma interessante alternativa para os dispositivos de interação atuais (teclado e mouse), na medida em que a interpretação dos gestos torna as interfaces computacionais mais fáceis e naturais de serem utilizadas (PAVLOVIC; SHARMA; HUANG, 1997).

Mas para utilizar corretamente gestos é necessário definí-los, diferenciando-os de postura para, assim, criar uma modelagem computacional dos gestos, de modo a utilizá-los no contexto desse projeto.

Tecnicamente, pode-se distinguir gesto e postura do seguinte modo (PAVLOVIC; SHARMA; HUANG, 1997):

Postura: Estado da mão e do antebraço em um instante de tempo, de modo que nesse estado pode-se definir claramente a posição e inclinação no espaço da mão, bem como o posicionamento dos dedos e da palma da mão.

Gesto: Variação da posição da postura no espaço em um determinado período de tempo. Neste caso, são avaliadas as posturas ou a postura da mão em uma trajetória em um período de tempo. De modo que, a mesma postura em trajetórias diferentes pode ser considerada como gestos diferentes, bem como a variação da postura na mesma trajetória também pode ser tratada como gestos diferentes.

Ou seja, deve-se tratar a postura de maneira estática e o gesto de modo dinâmico. Contudo ambos possuem uma interpretação bem definida.

Assim, tanto a postura como o gesto são considerados um meio de comunicação, como a fala, de modo que ao invés de vibrar as cordas vocais de modo a produzir som, é realizado um movimento com a mão e o braço de forma a compor uma comunicação visual, que o observador consegue interpretar.

Desse modo, Pavlovic; Sharma e Huang (1997) define o gesto como um "processo estocástico em um espaço de movimentos em um intervalo de tempo definido", ou seja, o gesto é a interpretação visual da movimentação do braço e da mão em um determinado cenário, sendo que não há gestos distintos para a mesma movimentação do braço e da mão em um mesmo ambiente de imagens. Por exemplo, apontar o dedo indicador para um computador e para uma pessoa tem interpretações distintas.

Tendo isso em mente, Pavlovic; Sharma e Huang (1997) também definem duas funções de gestos: a manipulativa e a comunicativa. Na primeira a movimentação do braço e da mão interage com um objeto real. Na segunda, o movimento por si só é suficiente para criar uma comunicação.

2.1.4 INTERAÇÃO EM AMBIENTES 3D

Na última década viu-se um crescimento de interfaces computacionais em três dimensões com o advento de hardware com maior capacidade

computacional e mais acessível, principalmente em jogos eletrônicos. E esse crescimento continua persistindo atuando também em áreas mais diversificadas, podendo citar como exemplo o XGL para Linux e o Second Life.

Essa utilização de sistemas em três dimensões aprimora não somente de maneira estética a interface, mas também a percepção na utilização da aplicação pelo usuário, já que na vida real as pessoas estão habituadas a um mundo em três dimensões (TORI, 2007).

Contudo, a manipulação de grande parte das interfaces em três dimensões ainda é realizada com controladores em duas dimensões, ou seja, em um plano, o que dificulta a utilização e compreensão do ambiente.

Recentemente, novas formas de interação com ambientes em três dimensões têm se popularizado, como o console de videogame Wii da Nintendo, cujo controle é um bastão sensível ao movimento o usuário, o que permite um menor tempo de aprendizagem do jogador, pois o modo de jogar simula um jogo real.

2.2 TECNOLOGIA

As principais ferramentas utilizadas neste trabalho são o HandVu, enJine e Java3D, sendo que todos estão altamente acoplados ao projeto.

2.2.1 HANDVU

O HandVu (pronuncia-se "hand view") é, até o momento, a biblioteca mais conhecida, de código aberto e gratuito que rastreia o movimento da mão e reconhece posturas determinadas no aplicativo em tempo real utilizando uma câmera, sem a necessidade de luvas especiais ou marcadores na mão, por isso sua utilização pode ser muito abrangente, explorando toda a capacidade do uso de gestos.

Essa biblioteca foi desenvolvida visando a confiabilidade da detecção de posturas e da inicialização rápida dos aplicativos em diversas condições ambientais e de luz (KOLSCH, 2004). Seu funcionamento baseia-se em reconhecer uma mão humana no vídeo, rastrear sua localização e reconhecer uma determinada configuração dos dedos (postura). Somente seis posturas podem ser reconhecidas e estão mostradas em na tabela 2.2.1.1.

		
Closed	LBack	LPalm
		
Open	Victory	Sidepoint

Tabela 2.2.1.1 – Posturas reconhecidas pelo HandVu

O HandVu possui um módulo principal que reúne métodos e algoritmos que garantem a robustez e desempenho em tempo real da detecção, rastreamento e reconhecimento de posturas. A execução do HandVu é dividida em duas fases:

Na primeira fase, é detectado a presença da mão em uma área limitada do vídeo e em uma postura particular, ao se realizar a detecção o HandVu é ativado. Nesse caso é utilizado a técnica de "multi-quadros". Essa técnica consiste em predeterminar um quadro onde o rastreamento da mão será iniciado. A detecção da mão é realizada verificando a diferença de cores entre o fundo do quadro e a mão. Após a inicialização o quadro é movido seguindo a mão do usuário (KOLSCH; TURK, 2004).

A segunda fase é responsável pelo reconhecimento da postura e rastreamento da mão. Para isso são utilizados quadros de imagens obtidas por fluxo óptico e distribuição probabilística de cores para rastrear rapidamente os movimentos da mão independente da postura da mesma. Nesse caso foi

utilizado o método de "Flock of Features" que consiste em capturar partes da mão, marcando-as na aplicação como pequenos quadros de imagem. Cada parte (feature) deve estar separada de outra, de modo a seguir uma distancia mínima e máxima. Além disso, cada feature é independente podendo se mover livremente, obedecendo às regras de distância, de modo a rastrear uma parte do objeto em questão, como os dedos da mão, por exemplo. Isso permite que o rastreamento seja flexível e veloz. Para o reconhecimento da postura foi utilizado o método Viola-Jones (VIOLA; JONES, 2003), que consiste em somar os elementos da imagem (feature), de modo que a partir de uma comparação entre essas somas é possível classificar cada elemento da imagem e a partir de uma classificação pré-determinada, é possível reconhecer o gesto do usuário (KOLSCH; TURK, 2004).

Contudo, sistemas divididos em fases são mais passiveis de propagação de erro e falhas em cada estágio. Para evitar esses problemas, cada estágio realiza uma estimativa da imagem, baseada em texturas em escala de cinza e informação local de cores de modo a minimizar os erros de detecção para aumentar a confiabilidade dos resultados.

2.2.2 ENJINE

O enJine é um game engine open-source implementado em Java que tem como objetivo ser uma ferramenta didática, como por exemplo para o ensino de computação gráfica (TORI et al., 2006) e também tem como objetivo facilitar a criação de ambientes tridimensionais em diversos trabalhos, principalmente para pesquisa de novas tecnologias. Baseado na API Java3D ela possui algumas características como:

- Renderização de ambientes 3D utilizando o Java3D;
- Detecção de colisões de múltiplos objetos;
- Criação facilitada de objetos do ambiente com visualização, colisão e alteração de estados através da utilização do core do enJine;
- Abstração das formas de input para o aplicativo.

Sua arquitetura é mais bem descrita em outro trabalho (NAKAMURA et al., 2006).

Esta última característica é a que merece maior atenção para este trabalho, uma vez que esta abstração, como ilustrado na figura 2.1.1.1, possibilita a integração de diferentes tipos de input com o enJine sem que seja necessário alterações do próprio código do pacote. Ou seja, esta arquitetura tenta desacoplar as relações das ações do jogo, representadas por objetos InputActrion, com os dispositivos de entrada, representados pela relação InputDevice e InputSensors. De modo que novos dispositivos podem ser adicionados estendendo esta relação.

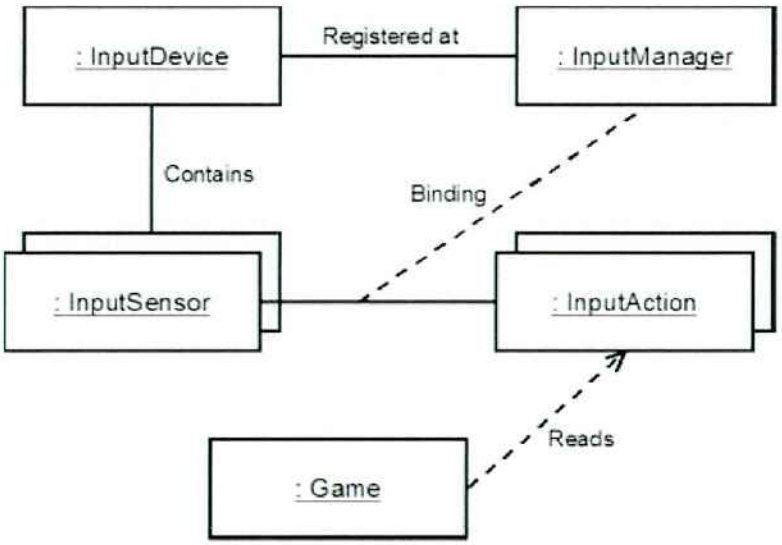


Figura 2.2.2.1 – Abstração do input na arquitetura do enJine

Alguns trabalhos como Robot ARena (CALIFE et al., 2007) , We AR Dancin (TSUDA et al., 2007) e WiimoteDevice (BERNARDES et al., 2007), utilizam esta estrutura para expandir os inputs do enJine utilizando a captura de imagens de uma câmera como entrada e as próprias imagens, para a criação de aplicativos em realidade aumentada no enJine.

No caso da integração do HandVu com a ferramenta, este aspecto facilita o trabalho, uma vez que esta se resume na criação de um InputDevice com os InputSensors para a utilização dos dados capturados pelo HandVu.

2.2.3 JAVA 3D

Java 3D é uma API de código aberto utilizada para desenvolvimento de aplicações em três dimensões, que consiste em uma biblioteca de classes para manipulação de objetos 3D e, como mostrado na figura 2.1.3.1, o Java3D é baseado nas bibliotecas OpenGL e DirectX. Para o desenvolvimento dos objetos é utilizado um grafo de cena que consiste em uma árvore de objetos que definem a aparência, geometria, orientação e localização dos objetos nas cenas criadas. A figura 2.1.3.2 apresenta um grafo de cena do Java3D (JAVA3D, 2007).

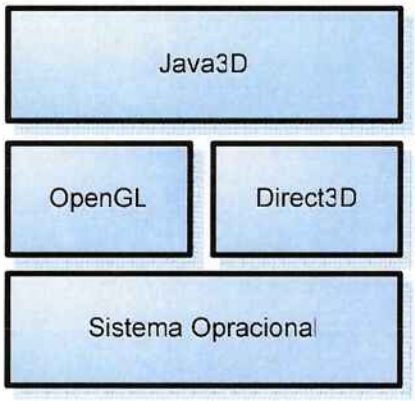


Figura 2.2.3.1 – Arquitetura de utilização do Java3D

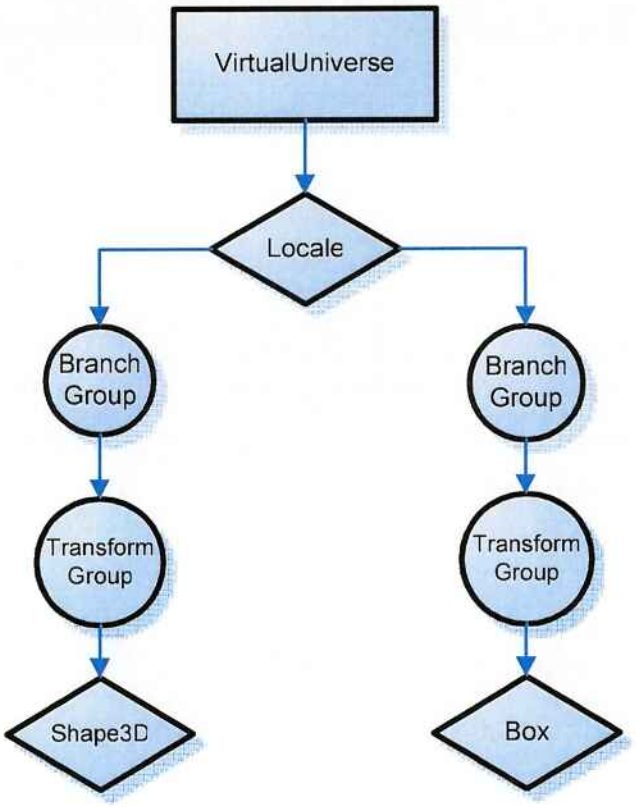


Figura 2.2.3.2 – Exemplo de grafo de cena.

3. ESPECIFICAÇÃO

A especificação do sistema consiste na abstração da implementação do sistema, descrevendo suas funções e requisitos. Também é realizada a análise do sistema utilizando a UML de modo a servir de guia para o desenvolvimento da aplicação.

3.1 DESCRIÇÃO DO SISTEMA

O Desktop3D consiste em uma mesa que contém blocos, que representam os arquivos e programas. Os blocos serão identificados na face superior com uma imagem representando o programa que abre o arquivo (essa imagem será a mesma que é utilizada nos ícones do sistema operacional Windows XP) e nas laterais pelo nome do arquivo, de forma semelhante a livros no ambiente físico. Os arquivos podem ser movidos dentro do ambiente de trabalho para qualquer lugar do espaço

De forma a organizar os arquivos no desktop, os mesmos podem ser empilhados, simulando uma pasta do Windows e há um modo para visualização do conteúdo da pilha devido a maior dificuldade em identificar rapidamente os arquivos da pilha, pois os ícones ficam sobrepostos sobre outro arquivo.

Também há um modo para rotação da câmera e mecanismos de *zoom in* e *zoom out* para facilitar a visualização completa do desktop.

3.2 REQUISITOS FUNCIONAIS

- Os arquivos manipulados são apenas os colocados pelo usuário na pasta de workspace, indicado ao inicializar o sistema.

- A interação com o usuário é apenas através de gestos e com auxilio do teclado para digitação de nomes ou teclas de atalho.
- A aplicação possui uma interface em três dimensões de modo a simular um ambiente real de organização de arquivos.
- Os arquivos são representados por blocos sendo a parte superior o ícone do arquivo e nas laterais o nome do arquivo.
- A câmera do ambiente pode ser movimentada pelo ambiente delimitado por uma área e ângulos, de modo que o usuário possa ter liberdade para visualizar qualquer espaço do ambiente.
- A dimensão do ambiente e dos arquivos é adequada, de modo que se possa ler o nome do arquivo sem a necessidade de aproximar a câmera.
- Os arquivos são movimentados livremente dentro do espaço do ambiente, inclusive para cima de outros arquivos.
- A resposta da interface com relação aos estímulos do usuário nunca deve demorar mais de 0,5 de modo a não dar a impressão que o aplicativo não está disponível. Em operação normal a interação deve ocorrer em tempo real, com atraso inferior a 0.2s (é difícil conseguir atrasos menores devido ao hardware de captura de imagem) e taxa de atualização de pelo menos 12 a 15 frames por segundo. Assim, parte do trabalho é levantar requisitos mínimos de hardware para que esses requisitos de tempo sejam atendidos.
- O ambiente simula um ambiente físico, ou seja, há detecção e tratamento de colisões e simulação dos efeitos da gravidade.
- Há um modo de visualização do conteúdo da pilha.
- Podem-se selecionar vários arquivos ou pilhas ao mesmo tempo.
- Pode se selecionar parte da pilha, de modo a retirá-la da mesma.
- O aplicativo utiliza o HandVu para captura, detecção de postura e de posicionamento da mão.

3.3 REQUISITOS NÃO FUNCIONAIS

- O aplicativo não é responsável por modificar fisicamente os arquivos como excluir ou mover para outros diretórios, mas essa funcionalidade poderá ser inclusa em uma versão futura.
- O aplicativo deve ser desenvolvido de modo modular e em camadas para garantir expansibilidade e escalabilidade, além de facilitar posteriores manutenções.
- O hardware requisitado pelo sistema deve ser de baixo custo, ou seja, webcams e GPUs de fácil acesso, custo inferior a R$ 300,00 (R$ 100,00 o custo de uma webcam e R$ 200,00 de uma placa de vídeo, no caso uma GEFORCE FX5500 256MB DVI/TV 8X AGP) e que normalmente possam ser utilizados para outras aplicações, como jogos, CAD, chats online, entre outros.

3.4 CASOS DE USO

A seguir é apresentado o diagrama de casos de uso na figura 3.5.1. Nesse diagrama o ator, que é sempre o mesmo, o usuário do sistema, foi suprimido do diagrama para não poluí-lo desnecessariamente:

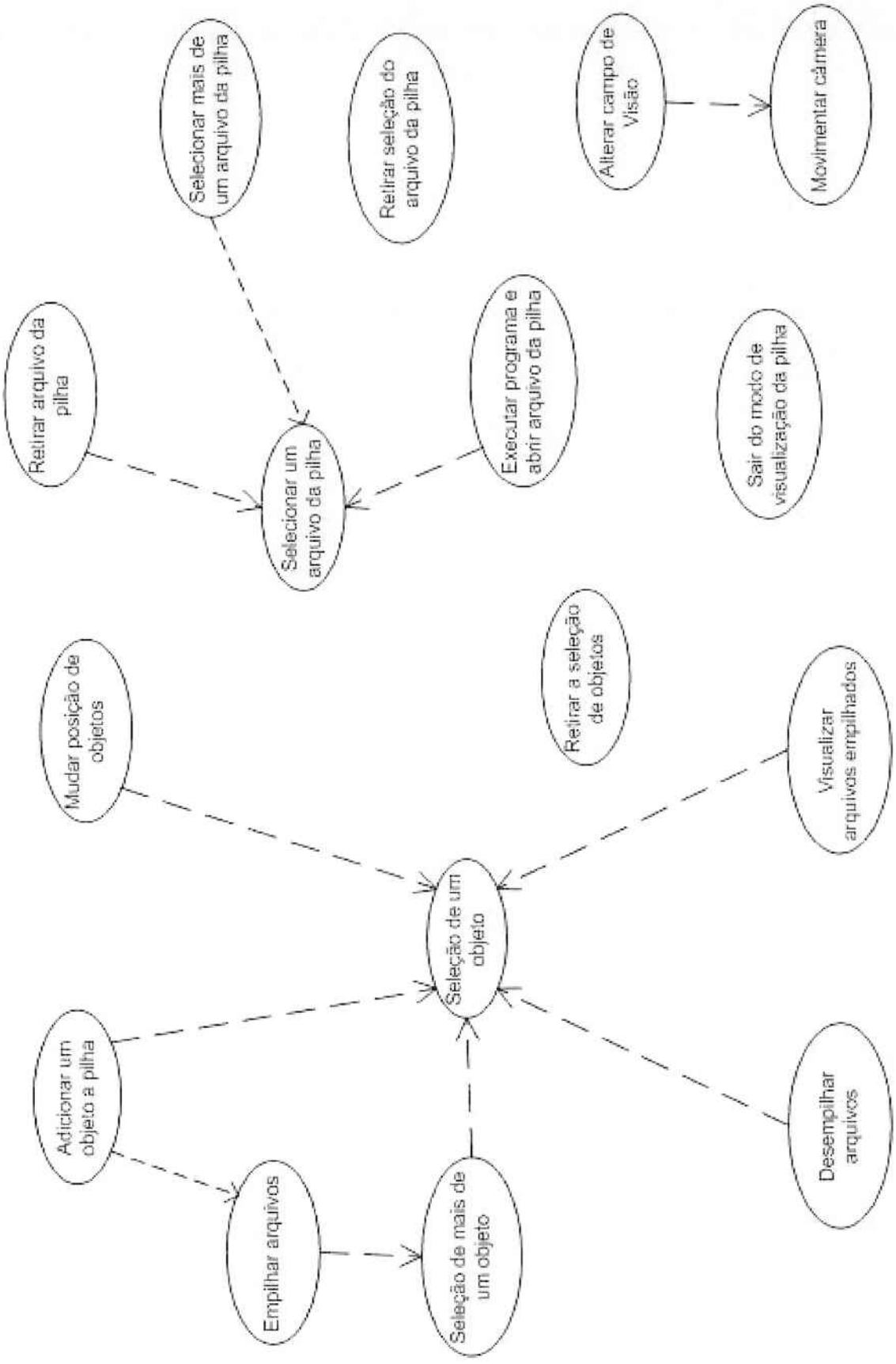


Figura 3.4.1 – Diagrama de Casos de Uso

3.5 DIAGRAMA DE CLASSES

O diagrama de classe do sistema é apresentado na figura 3.6.1

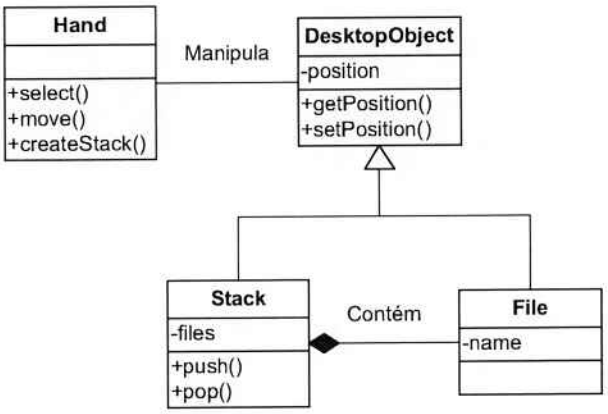


Figura 3.5.1 – Diagrama de classe do Desktop3D

Nesse diagrama nota-se a comunicação entre os objetos principais da aplicação, que consiste na manipulação de um objeto do desktop pela mão, sendo possível selecionar um objeto, mover ou empilhar arquivos.

Os objetos do desktop podem ser arquivos ou pilhas, sendo que a pilha contém uma coleção de arquivos.

Esta abordagem permite abstrair as funções dos objetos do desktop facilitando a manutenção de operações comuns a pilhas e arquivos.

3.6 ARQUITETURA

A arquitetura do sistema foi planejada baseada na plataforma do enJine, utilizando a abstração de entrada de dados pelo usuário e o encapsulamento das funções do Java3D. A figura 3.7.1 apresenta o diagrama da arquitetura.

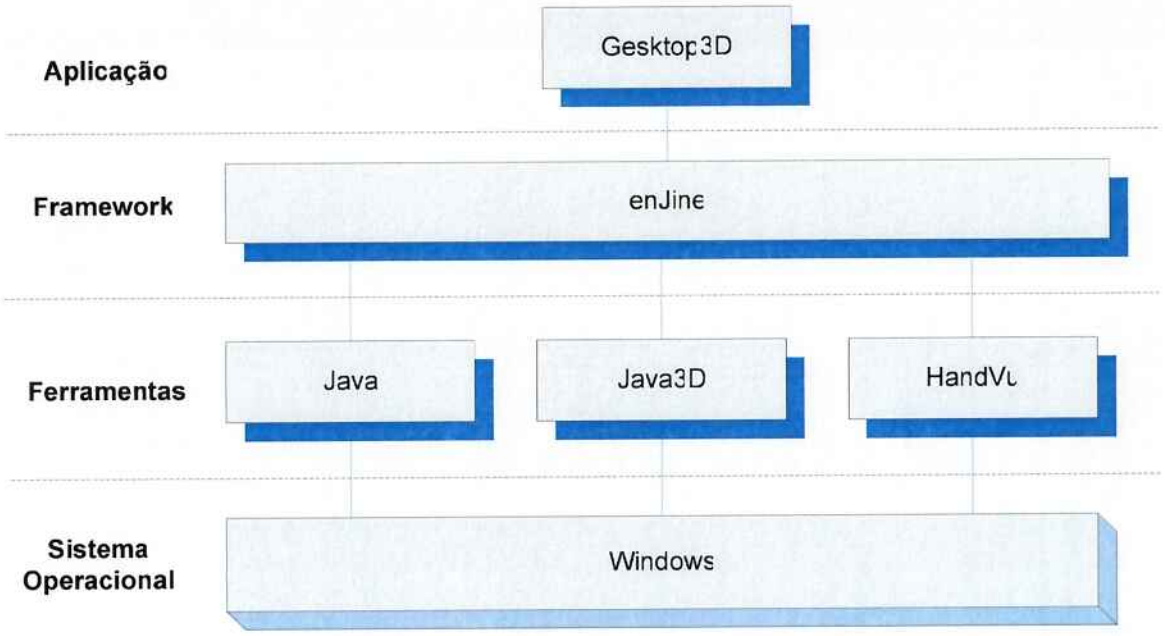


Figura 3.6.1 – Diagrama de camadas do sistema

Pelo diagrama de arquitetura nota-se a divisão de camadas do sistema, explicadas a seguir:

- Sistema operacional – O sistema operacional que serve de base para o sistema é o Microsoft Windows, de modo que o Desktop serve como a interface entre o usuário e o sistema operacional, fornecendo funções para organização e manipulação de arquivos.
- Ferramentas – Utilizou-se como ferramenta de desenvolvimento o Java e Java3D devido à utilização do enJine. Também foi utilizado o HandVu como a aplicação de rastreamento da mão do usuário e reconhecimento de gestos.
- Framework – O enJine foi utilizado como framework de trabalho devido ao encapsulamento do HandVu, Java3D e demais funções, como manipulação de objetos em três dimensões e reconhecimento de colisão entre objetos.
- Aplicação – A última camada diz respeito ao Desktop3D que se resume ao núcleo principal da aplicação, a interface ao usuário e manipulação dos arquivos.

Assim, essa arquitetura ilustra a divisão das funções por camadas, de modo a demonstrar a organização do sistema, indicando a responsabilidade de

cada componente. É necessário ressaltar que, embora a arquitetura apresentada possa ser replicada para outras aplicações, o sistema possui os componentes altamente acoplados, devido à alta dependência da aplicação aos outros componentes, por isso não será possível substituir os componentes das camadas superiores.

3.7 RECURSOS E INFRA-ESTRUTURA

Nesse trabalho foram utilizados os seguintes softwares:

- Sistema operacional: Windows XP;
- OpenCV versão Beta 5;
- HandVu versão beta 3.

O hardware mínimo necessário para o aplicativo é:

- Computador com um processador de no mínimo 1.5GHz;
- Câmera com resolução superior a 320x240, sendo de USB ou firewire e dando suporte ao libdc 1394.

O sistema também utilizará o game engine enJine, desenvolvido pelo InterLab da USP, necessitando também a instalação do JRE e a instalação do JAVA3D.

4. METODOLOGIA

Neste projeto se utiliza a metodologia do Processo Unificado, uma vez que os autores possuem maior familiaridade com o processo e suas ferramentas. Também, devido ao pequeno número de membros no projeto e pouco tempo para a criação (três membros e prazo de entrega de aproximadamente 12 meses) a flexibilidade da metodologia facilita o desenvolvimento devido à diminuição do tempo de sua aplicação, alem de reduzir os erros e re-implementações.

No processo unificado o projeto pode ser separado em 4 fases, que possuem diferentes focos. Estes são Concepção, Elaboração, Construção e Transição e cada fase pode ser separada em várias iterações (JACOBSON; BOOCH; RUMBAUGH, 1998). Cada iteração tem como resultado em um incremento na funcionalidade do sistema. E cada iteração pode, por sua vez, ser separada em etapas de captura de requerimentos, análise, projeto, implementação e testes.

Em caso de pequenos projetos a metodologia pode ser adaptada de forma que as fases de concepção e elaboração podem ser descartadas, como proposto por Smith (2002).

Nesse projeto, se seguiu essa customização da metodologia, correndo rapidamente por essas fases. Desse modo o projeto se foca mais na fase de construção e em três iterações: a da criação de mini-protótipos de forma a implementar e/ou testar pequenos módulos do aplicativo final, a integração do HandVu e do enJine e a implementação do aplicativo final.

Na iteração da implementação dos mini-protótipos o grupo pode ser divido de forma que cada membro desenvolva um protótipo. Na iteração da implementação final o grupo pode ser dividido de modo que cada membro se encarregue da integração das funcionalidades testadas com os mini-protótipos.

5. PROJETO E IMPLEMENTAÇÃO

Seguindo a metodologia apresentada foram desenvolvidos protótipos como prova de conceito e, posteriormente, foram criados paralelamente o Desktop3D e a integração do HandVu com o enJine.

5.1 PROTÓTIPOS

Como forma de prova de conceito e para estudar o HandVu, foram desenvolvidas aplicações de teste das tecnologias utilizadas, apresentadas a seguir.

5.1.1 UTILIZAÇÃO DO HANDVU NO JAVA POR SOCKETS

O primeiro protótipo criado foi a captura das informações do HandVu no Java através de sockets. O HandVu é uma aplicação de rastreamento da mão e reconhecimento de posturas pré-determinadas, e todas essas informações são colocadas automaticamente pelo HandVu na porta 7045, assim o HandVu funciona como um "servidor" de reconhecimento de gestos. Com isso, foi desenvolvida uma aplicação em Java que se conecta a porta 7045 através de sockets, recupera as informações enviadas e as utiliza de maneira adequada. Contudo, para essa abordagem é necessário possuir dois processos executados no sistema, o HandVu (Servidor) e a aplicação (Cliente). A figura 5.1.1.1 apresenta uma figura da utilização do protótipo.

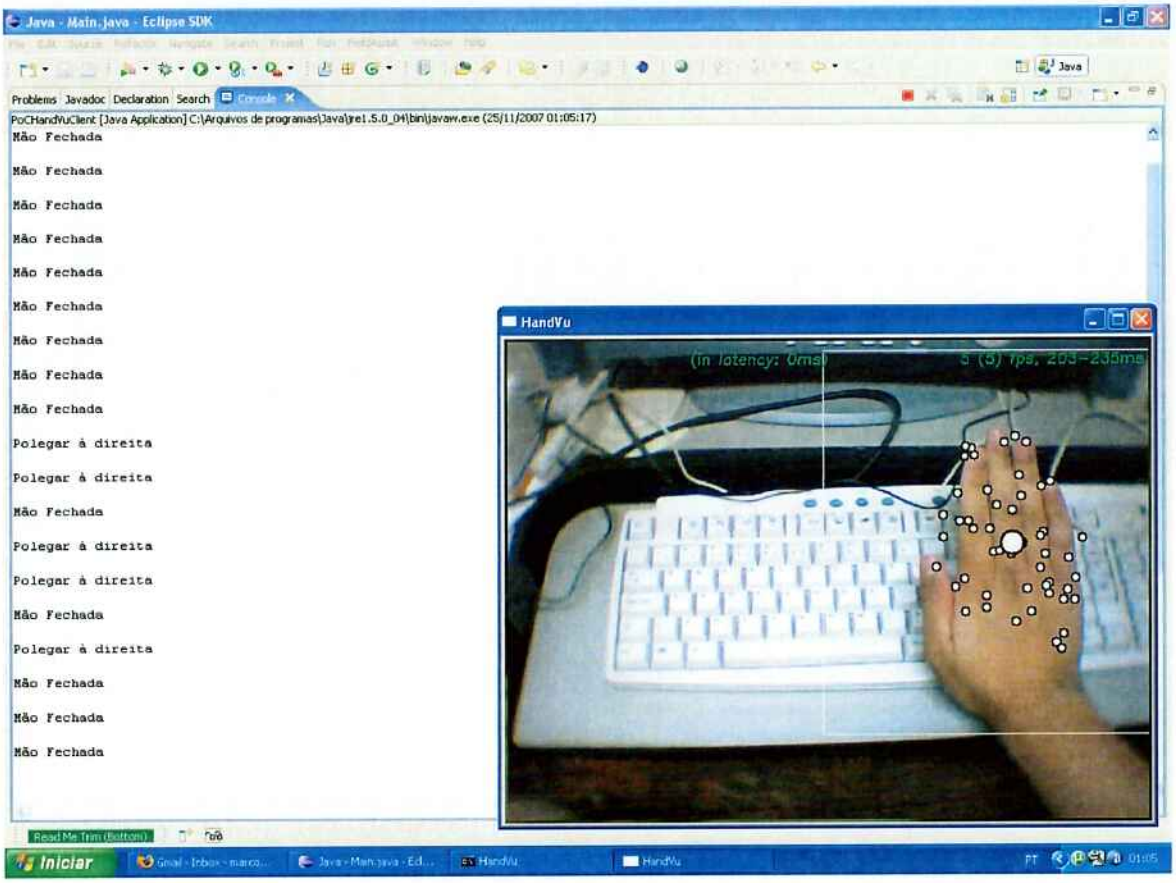


Figura 5.1.1.1 – Captura das informações do HandVu através de sockets.

5.1.2 ESTUDO E COMPILAÇÃO DO HANDVU

Como parte do estudo do HandVu seu código-fonte foi analisado e uma aplicação de teste foi desenvolvida. O HandVu foi desenvolvido utilizando a biblioteca OpenCV versão beta 5. As principais funções e estruturas da aplicação são descritas a seguir:

- hvState – Estrutura de dados com as informações do reconhecimento do HandVu, possui os seguintes parâmetros:
  - obj_id – Identificação da mão utilizada, o valor padrão utilizado é zero, indicando a utilização da mão direita.
  - tracked – Indica se a mão do usuário está sendo rastreada.
  - recognized – Indica se o HandVu conseguiu detectar uma postura.

  - center_xpos – Indica a posição da mão referente ao eixo X com relação a janela, seu valor é um número real que varia de zero a um.
  - center_ypos – Análogo ao parâmetro center_xpos, mas com relação ao eixo Y.
  - scale – Escala da mão rastreada.
  - posture – Nome da postura reconhecida.
  - tstamp – Tempo em milisegundos de reconhecimento da mão.
- hvGetState – Função que retorna um objeto hvState com as informação capturadas pelo HandVu.
- hvInitialize – Função que inicializa o HandVu com as dimensões da janela utilizada, reconhece a câmera utilizada e prepara a área de reconhecimento da mão.
- hvLoadConductor – Carrega as informações do condutor, que consiste em um arquivo com parâmetros para reconhecimento da mão, como a calibração da câmera, área de reconhecimento e arquivos com as máscaras de cada postura.
- hvStartReconigtion – Ativa os scanners de detecção de postura do HandVu baseado no condutor utilizado.
- hvStartGestureServer – Cria um "servidor", enviando as informações da estrutura hvState na porta desejada.

5.2 DESKTOP 3D

Após o desenvolvimento dos mini-protótipos e elaboração dos casos de uso, foi possível conceber a arquitetura do Desktop3D, chamado de Gesktop3D (Gesture-based Desktop3D) e também ilustrar a parte gráfica da aplicação como mostrado nas figuras 5.2.1 e 5.2.2.

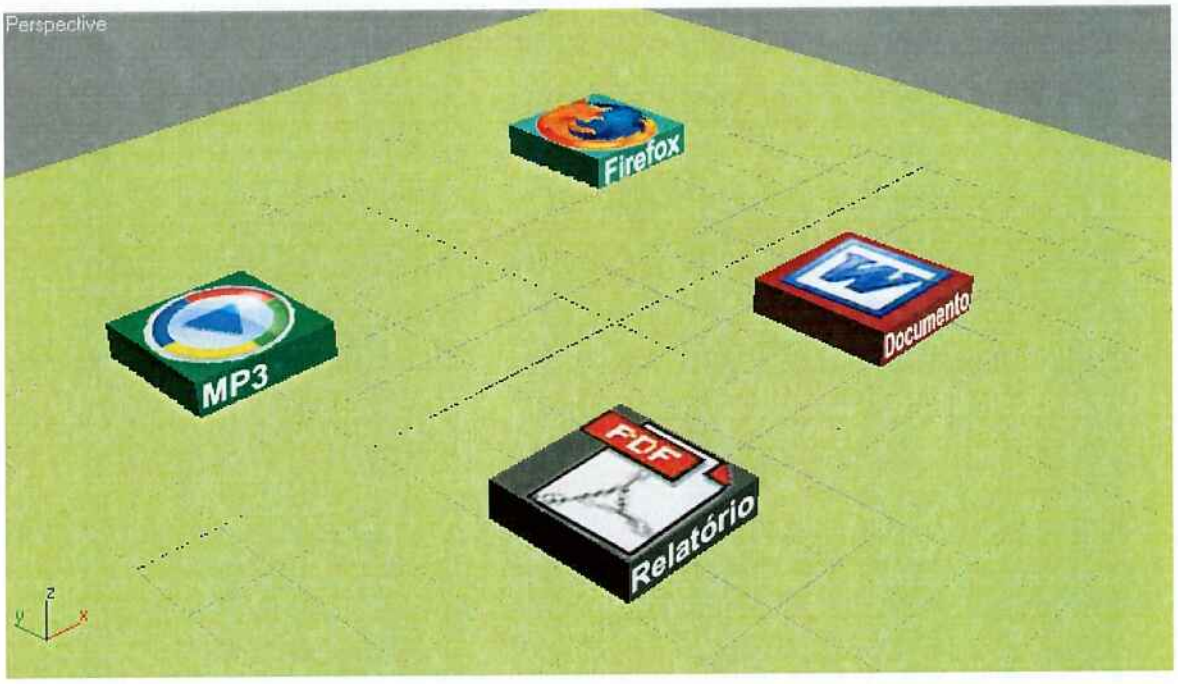


Figura 5.2.1 – Concepção dos arquivos no Desktop3D

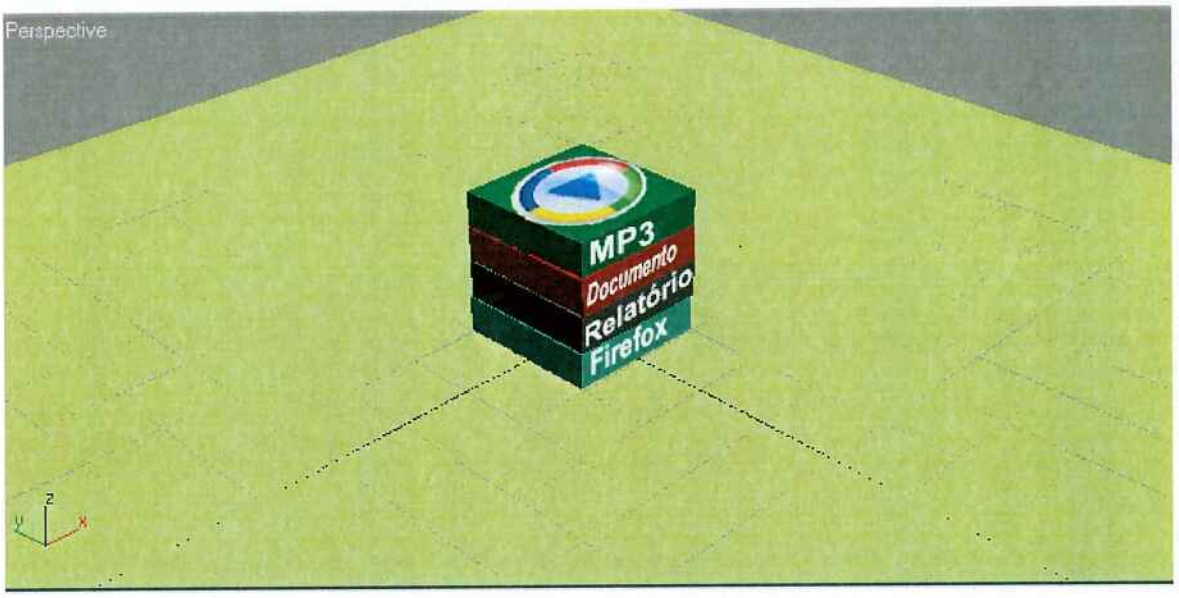


Figura 5.2.2 – Concepção dos arquivos organizados em pilha.

Partindo dessa concepção, deu-se inicio à fase de analise da aplicação. Para isso, utilizamos a UML (Unified Modeling Language), uma ferramenta padronizada de modelagem de sistemas orientados a objetos (FOWLER, 2005). Assim, foram geradas a arquitetura do sistema com um diagrama de componentes e a arquitetura interna do Gesktop3D utilizando-se um diagrama de classe e pacote.

5.2.1 ARQUITETURA INTERNA

A arquitetura do Gesktop3D foi desenvolvida baseada na funcionalidade de cada objeto dentro do desktop, assim é possível delimitar o papel de cada classe no sistema atribuindo para cada objeto suas funções de comportamento e visualização. A figura 5.2.2.1 apresenta o diagrama de classe e pacotes do Gesktop3D, de modo a ilustrar a arquitetura interna da aplicação.

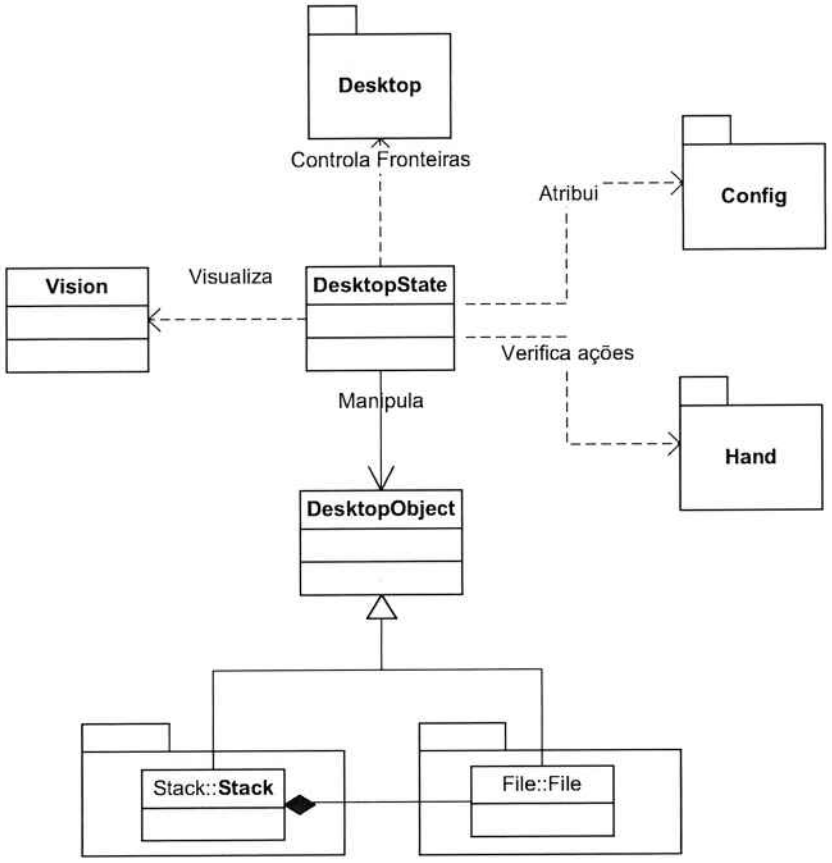


Figura 5.2.1.1 – Arquitetura interna do Gesktop3D.

A seguir são apresentadas as funções de cada classe ou pacote:

- DesktopState – Classe principal do desktop, sendo responsável por determinar a interface de interação (teclado, mouse ou HandVu) e aplicar a cada sensor sua respectiva classe de ação (GameAction), por exemplo

a ação "Move Forward" é ligada à seta "Para cima" do teclado. Esse pacote também é responsável por iniciar as configurações do sistema e chamar os métodos de criação de objetos na cena.

- Desktop – O pacote Desktop possui as classes de manipulação e visualização da mesa, onde são colocados os objetos do desktop.
- Vision – Vision é a classe responsável pelo controle de visualização da mesa, como movimentação de câmera e funções de zoom in e zoom out.
- Config – Esse pacote concentra as classes de configuração do sistema e de ações sobre o sistema. As classes desses pacotes são acessadas a partir de qualquer outro objeto da aplicação.
- Hand – O pacote hand possui as classes para manipulação dos objetos do desktop e também o modo de visualização da mão na mesa.
- DesktopObject – Essa classe é utilizada como abstração dos objetos da mesa, pois algumas funções de manipulação são comuns e aplicáveis a todos os objetos, assim é possível encapsular os métodos comuns em uma única classe e replicá-los aos demais objetos.
- File – O pacote File possui as classes de manipulação e visualização de arquivos do desktop.
- Stack – Esse último pacote apresenta as classes de controle de pilhas de arquivos do desktop que se caracteriza, basicamente, por uma lista de arquivos que podem ser controlados simultaneamente.

Ainda há um segundo modo de utilização do Gesktop3D, o modo de visualização de arquivos da pilha, que permite manipular os arquivos da pilha isoladamente. A arquitetura do módulo de visualização é apresentada na figura 5.2.1.2, e a seguir são descritos os pacotes utilizados.

ViewStackState – De forma análoga à classe DesktopState, a ViewStackFile é a principal classe do modo, sendo responsável por aplicar as configurações iniciais, organizar a visualização dos arquivos, aplicar as ações aos sensores correspondentes e controle de manipulação dos arquivos, que consiste em selecionar arquivos para serem retirados da pilha ou executar um arquivo.

Config – Esse pacote é o mesmo da arquitetura interna, assim possui as mesmas funções.

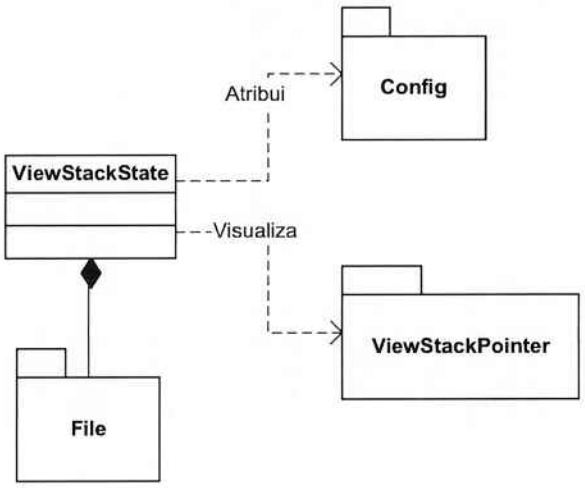


Figura 5.2.1.2 – Arquitetura do modo de visualização de arquivos da pilha.

ViewStack – Esse pacote possui a classe de visualização do Desktop, na implementação é utilizado para acrescentar um papel de parede ao modo de visualização.

File – Consiste no mesmo pacote da arquitetura anterior, contudo apenas as funções de visualização dos arquivos são utilizadas.

Assim, percebe-se que no modo de visualização de arquivos toda a lógica de negócio é acoplada ao pacote principal.

5.2.3 DESENVOLVIMENTO

Terminada a concepção da arquitetura, deu-se início ao desenvolvimento do Desktop3D, implementando cada classe e pacotes apresentados. Os principais pacotes do sistema (Desktop, Hand, Stack, File e ViewStackPointer), seguem a estrutura de objetos de jogo (GameObjects) do enJine, ilustrado na figura 5.2.3.1.

Estas classes são utilizadas para desacoplar funções distintas do objeto manipulado, como descritas a seguir:

GameObject – Classe principal da estrutura, possui toda a lógica de negocio do objeto em questão.

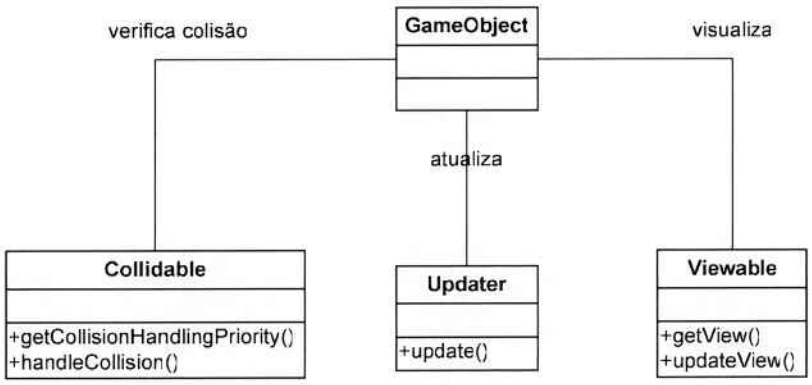


Figura 5.2.3.1 – Diagrama de classe da estrutura de objetos do enJine.

Collidable – Essa classe é responsável pela lógica de colisão de objetos.

Updater – Classe com funções de atualização do objeto, sendo o método update chamada por uma freqüência determinada no jogo.

Viewable - Classe com a estrutura de visualização do objeto, que consiste em um ramo do grafo de cena da aplicação.

Baseado nesses conceitos as classes de cada pacote foram desenvolvidos, suas funções são explicadas a seguir:

5.2.3.1 PACOTE CONFIG

GameConfig – Classe com as configurações do sistema como o tipo de interface utilizada (teclado, mouse ou HandVu) e estados do sistema (MainGameMode, CameraMode ou StackViewMode). Também possui os valores comuns e constantes do sistema, como a posição inicial da câmera, a dimensão da mesa, dos arquivos e da mão, a velocidade de movimento dos arquivos, aceleração da gravidade, número da prioridade de cada colisão, sensibilidade de movimentação da mão no HandVu e código de cada postura.

GameActions – Classe que contém todas as ações do Desktop3D (InputAction do enJine) que compreendem as ações de movimentação com o

teclado e com o mouse, ação de mudança de modo de utilização, seleção de objeto e criação de pilha.

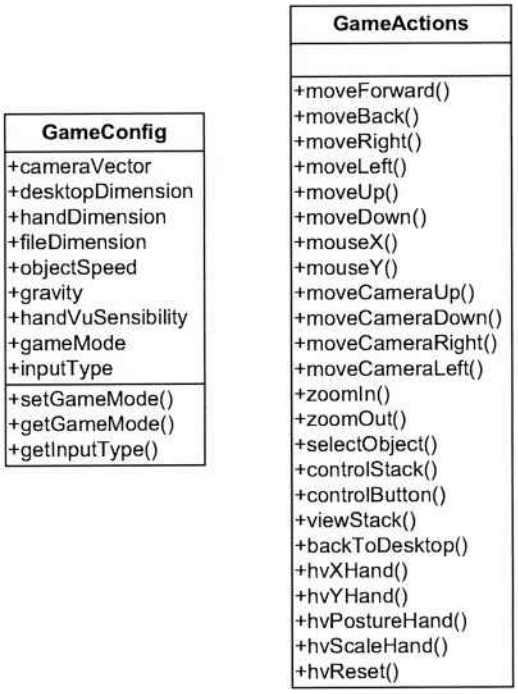


Figura 5.2.3.1.1 – Diagrama e classe do pacote Config.

5.2.3.2 PACOTE DESKTOP

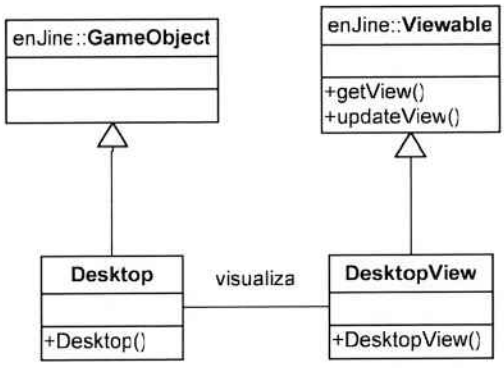


Figura 5.2.3.2.1 – Diagrama de classe do pacote Desktop

Desktop – Classe utilizada para manter uma instância da classe DesktopView.

DesktopView – Classe utilizada para gerar a visualização da mesa na aplicação.

5.2.3.3 PACOTE HAND

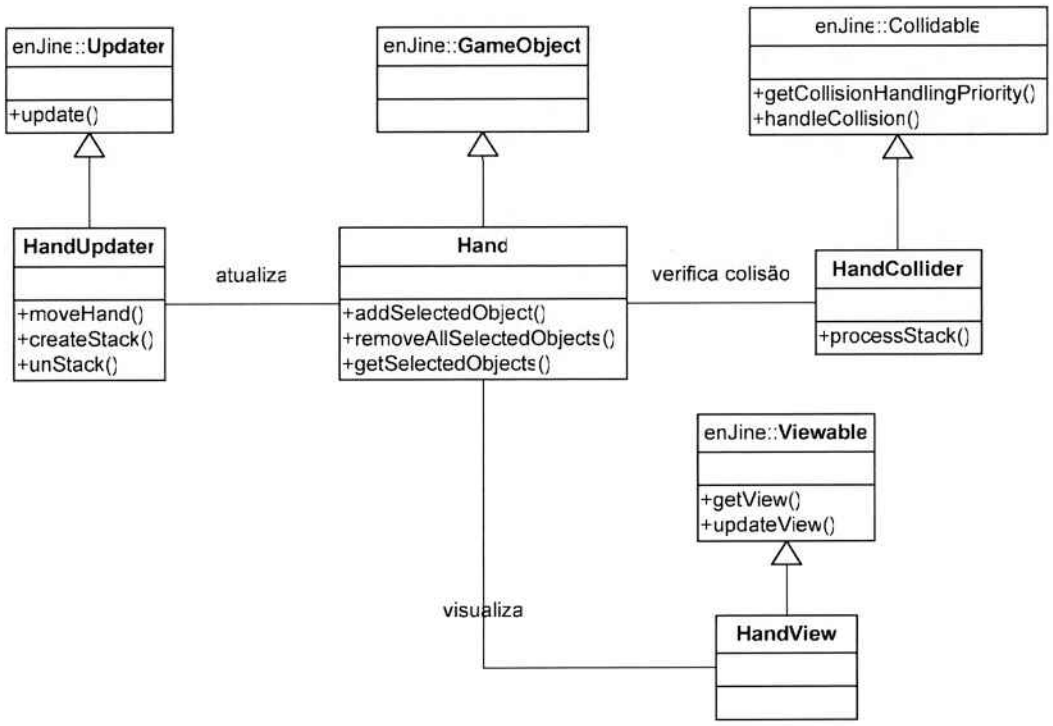


Figura 5.2.3.3.1 – Diagrama de classe do pacote Hand.

Hand – Classe responsável por manter a posição da mão no desktop e gerenciar os objetos selecionados.

HandUpdater – Classe que realiza as ações da mão, possui as funções de movimentação da mão pela mesa, criação de pilha e desempilhamento dos arquivos.

HandView – Essa classe é utilizada para gerar a visualização da mão na aplicação.

HandCollider – Delimita os limites de colisão da mão, realiza a movimentação da mão e seleciona objetos da mesa.

5.2.3.4 PACOTE FILE

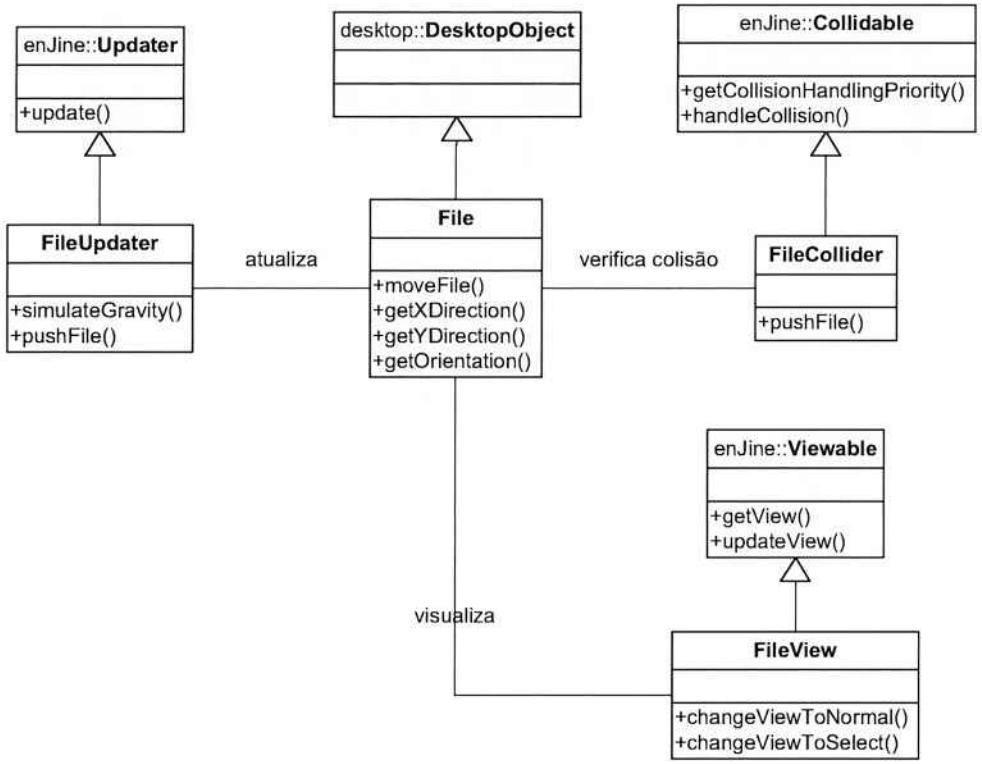


Figura 5.2.3.4.1 – Diagrama de classe do pacote File

File – Essa classe possui as informações de posicionamento do arquivo, atributos específicos, como nome, autor e tamanho. Também é responsável por movimentar o arquivo e definir a direção de movimentação.

FileUpdater – Classe responsável por simular a gravidade dos arquivos na mesa e empurrar o arquivo caso outro objeto colida com o primeiro.

FileView – A classe possui a função de geração da visualização do arquivo e modificar a cor do arquivo, caso esse esteja selecionado.

FileCollider – Verifica se houve uma colisão com outro objeto, e, nesse caso, chama as funções necessárias para "separar" os arquivos.

5.2.3.5 PACOTE STACK

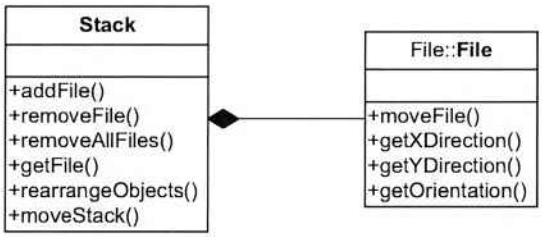


Figura 5.2.3.5.1 – Diagrama de classe do pacote Stack

Stack – Classe responsável pelo controle de arquivos organizados em uma mesma pilha.

No caso da pilha, não foram desenvolvidas as classes Updater, Viewable e Collidable, pois essas seriam utilizadas da classe dos arquivos contidos na pilha.

5.2.3.6 PACOTE VIEWSTACKPOINTER

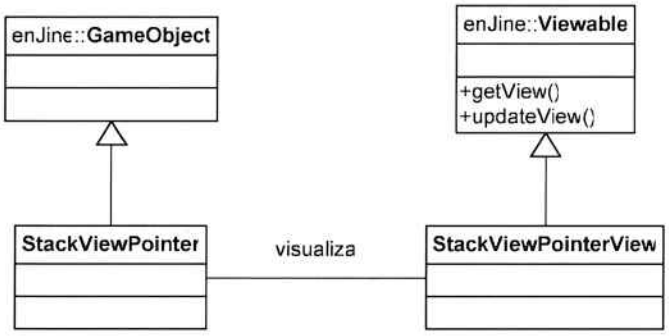


Figura 5.2.3.6.1 – Diagrama de classe do pacote ViewStackPointer

StackViewPointer – Classe utilizada para manter uma instância da classe StackViewPointerView.

StackViewPointerView – Classe que contém a geração do plano de fundo para o modo de visualização de arquivos da pilha.

5.2.4 UTILIZAÇÃO DO SISTEMA

Depois de descrito a arquitetura do sistema e apresentado o modelo estático, descreve-se a seguir a utilização do Gesktop3D, de modo a indicar o funcionamento e dinâmica da aplicação.

O desktop utiliza necessariamente a estrutura de bibliotecas e diretórios apresentados na figura 5.2.4.1.

Figura 5.2.4.1 – Estrutura de arquivos do Gesktop3D

A biblioteca Gesktop3D possui as classes da aplicação, Já os pacotes j3dcore, j3dutils e vectmath são as bibliotecas fornecidas pelo Java3D, nesse caso utilizamos a versão 1.5.1. O pacote enJine refere-se ao enJine utilizado. Foi utilizado a versão 4.0 alpha do enJine, além disso, foi encontrada uma falha no código-fonte, que foi consertada pelo grupo durante o desenvolvimento do Gesktop3D. O enJine possuía um problema na classe Mouse do pacote interlab.enjine.io.input, nos métodos mousePressed e mouseReleased, ambos utilizavam o método getID da classe MouseEvent para verificar o botão do mouse pressionado. Contudo para essa verificação deve ser utilizado o método getModifiers da mesma classe. O problema foi corrigido e reportado aos responsáveis pelo enJine.

A biblioteca HandVu_JNI é a interface entre o HandVu e o Java, utilizando JNI, o pacote HandVuDevice possui a classe HandVuDevice que estende a classe InputDevice do enJine, de modo a utilizar o HandVu como um dispositivo de entrada para a aplicação. De maneira análoga, funciona o pacote HandVuSocketDevive, essa última recebe as informações do HandVu através de sockets ao invés de JNI, como o pacote anterior.

Estabelecida a estrutura, pode-se utilizar o Gesktop3D. Há quatro modos de inicialização da aplicação, sendo necessário utilizar o seguinte comando seguido de um parâmetro, indicando o modo de utilização:

java –jar Gesktop3D.jar –O

Sendo O uma das opções da tabela 5.2.4.1.

Opção (Parâmetro O)	Modo
k	Interação através de teclado
m	Interação através de mouse e teclado
hj	Interação através do HandVu com JNI
hs	Interação através do HandVu utilizando *Sockets*

Tabela 5.2.4.1 – Modos de operação do Gesktop3D.

Para a última opção é necessário iniciar o HandVu, para esse ser utilizado como um servidor de gestos e posicionamento da mão.

A utilização do Gesktop3D é análoga ao desktop do Microsoft Windows, o usuário movimenta o cursor pela mesa e manipula os objetos, sendo possível selecionar arquivos e deslocá-los livremente no espaço sobre a mesa, inclusive levantar o arquivo da mesa. Há mecanismos de detecção de colisão que não permitem que arquivos distintos ocupem o mesmo espaço, assim quando um arquivo colide com outro, o ultimo é empurrado seguindo a mesma direção de movimento do primeiro arquivo. Também há mecanismos de simulação gravitacional que verificam a altura do arquivo e o movimento com aceleração constante caso esse não esteja sobre o desktop. A figura 5.2.4.2 indica o plano de coordenadas utilizado na aplicação.

Figura 5.2.4.2 – Visualização geral do Gesktop3D e seu plano de coordenadas.

A fim de melhorar a organização da mesa é possível empilhar arquivos seguindo critérios escolhidos pelo usuário como mesmo tipo de arquivo, músicas, por exemplo, ou dentro de um mesmo contexto como arquivos da monografia que compreendem arquivos de texto, diagramas, imagens e cronograma. Para empilhar o usuário seleciona os arquivos desejados e aciona o comando necessário de acordo com o modo de entrada de dados utilizado. A aplicação posiciona a pilha no baricentro da forma geométrica cujos vértices são as posições dos arquivos selecionados. A aplicação empilha os arquivos na ordem de seleção dos mesmos, também é possível criar uma pilha a partir de uma ou mais pilhas, nesse casso segue-se do mesmo modo, seleciona-se e faz-se o comando de empilhar, a ordem dos arquivos na nova pilha também segue a ordem de seleção. A figura 5.2.4.3 apresenta os arquivos selecionados que serão empilhados.

Figura 5.2.4.3 – Arquivos selecionados que serão empilhados.

Como os arquivos, a pilha também pode ser movimentada livremente sobre o desktop, contudo, não é possível levantá-la. A característica que diferencia o deslocamento da pilha com relação ao arquivo é que a velocidade do movimento é inversamente proporcional a quantidade de arquivos na pilha, ou seja, quanto mais arquivos empilhados, mais difícil se torna mover a pilha para outra posição. Além disso, para aproximar a pilha da realidade física, os arquivos ao serem empilhados não ficam alinhados, cada arquivo é rotacionado no eixo z em um ângulo entre zero e dois radianos. A figura 5.2.4.4 apresenta uma pilha criada a partir dos arquivos selecionados na figura 5.2.4.2.

Figura 5.2.4.4 – Pilha criada

Para auxiliar o usuário a utilizar a interface em três dimensões e visualizar os objetos da mesa, é possível controlar a câmera do Gesktop3D. Para isso, o usuário realiza o comando de mudança de controle de acordo com o modo de interação utilizado. Assim é possível movimentar a câmera para os lados, para cima e para baixo e efetuar zoom in e zoom out, essas funções também dependem do dispositivo de interação utilizada. A figura 5.2.4.5 apresenta o desktop visto por outro ângulo.

Figura 5.2.4.5 – Variação de visualização do desktop.

Depois de a pilha ser criada, torna-se difícil visualizar pelo desktop o conteúdo da mesma, por isso, foi desenvolvido o modo de visualização de arquivos da pilha. Dependendo da interface de interação utilizada, o usuário realiza o comando especifico para acionar o modo de visualização. Nesse modo, os arquivos são apresentados de forma que o ícone fique de frente para o usuário e são organizados de modo a gerar uma circunferência, onde o ícone que poderá ser manipulado ficará no centro. Assim, há sempre um arquivo focado que fica no centro da tela, os demais são rotacionados à 20º para que o usuário possa visualizar quais são esses arquivos. Além disso, para o arquivo focado são apresentadas suas características como o nome, autor e tamanho do arquivo em MegaBytes em um frame na parte inferior da tela. Nesse modo não é possível alterar o posicionamento da câmera. A figura 5.2.4.6 apresenta o modo de visualização de arquivos.

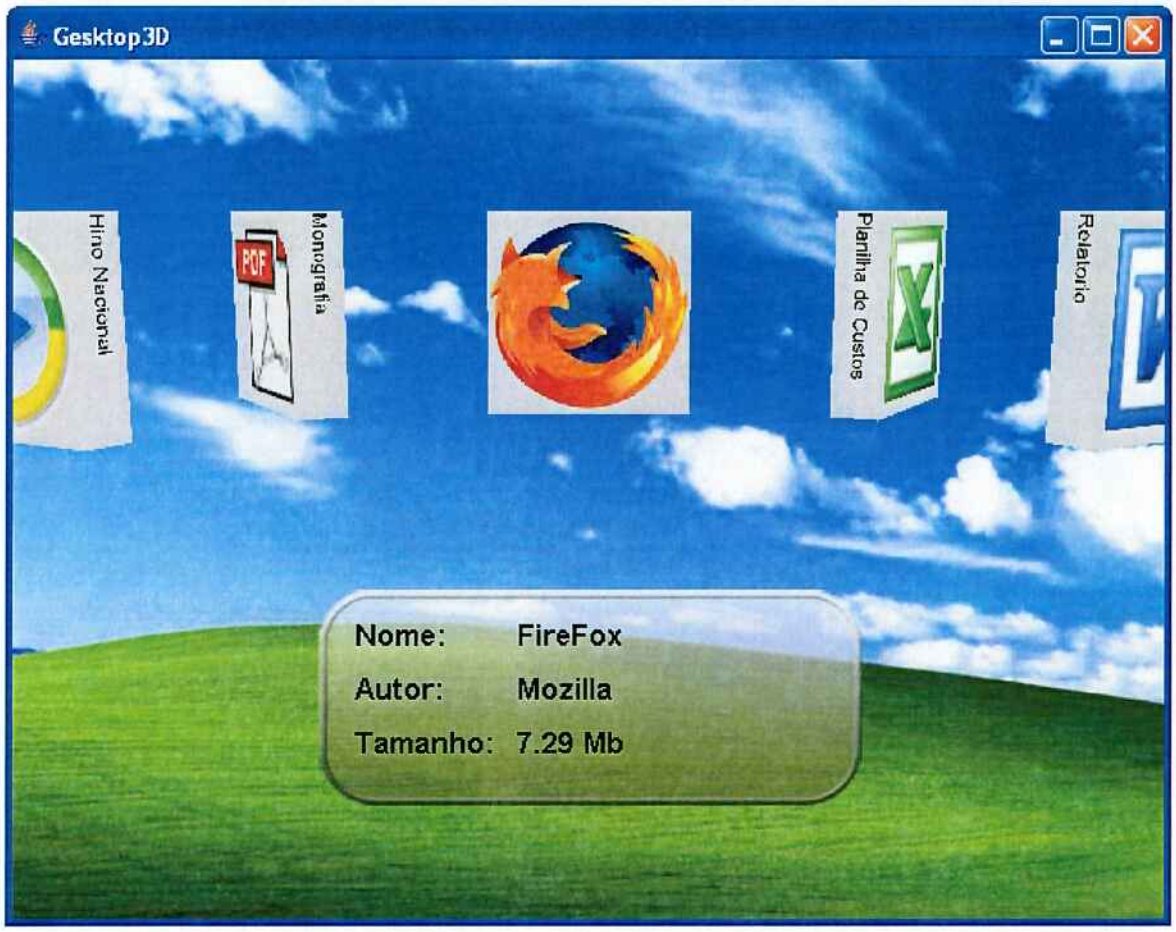


Figura 5.2.4.6 – Modo de visualização de arquivos.

No modo de visualização de arquivos, além de obter as informações dos arquivos da pilha, é possível retirar os arquivos da mesma, para isso é necessário selecionar os arquivos que serão retirados e realizar um comando especifico para retornar ao desktop. Os comandos para seleção de arquivos e para sair do modo de visualização dependem do dispositivo de interação utilizado. A figura 5.2.4.7 apresenta os arquivos da pilha selecionados e a figura 5.2.4.8 o modo desktop com os arquivos retirados da pilha.

Voltando ao modo desktop é possível desempilhar todos os arquivos da pilha, para isso há um comando especifico que depende do dispositivo interativo utilizado.

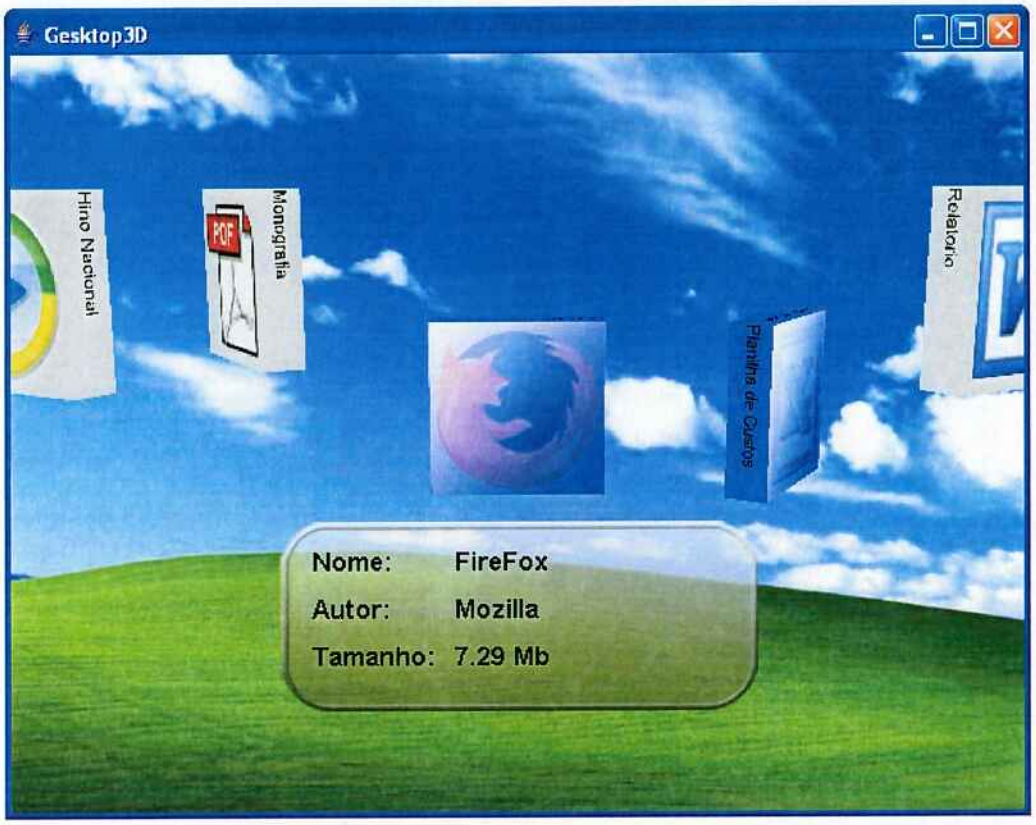


Figura 5.2.4.7 – Seleção dos arquivos da pilha

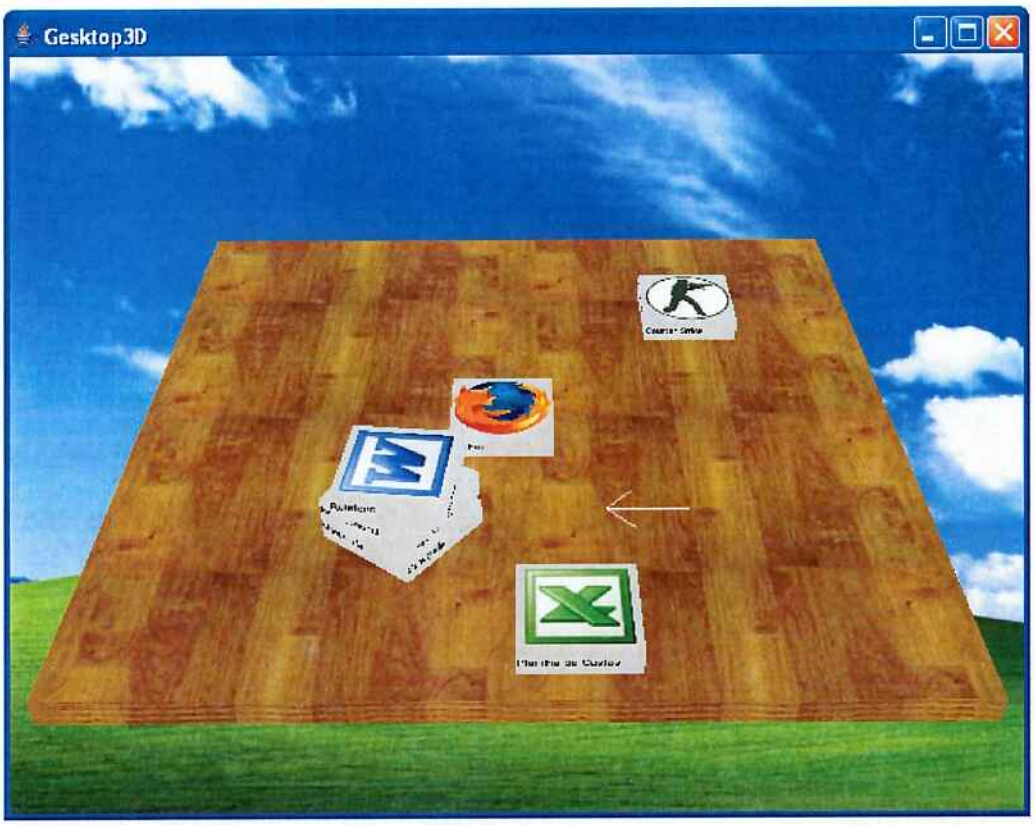


Figura 5.2.4.8 – Modo desktop com os arquivos retirados.

5.2.5 COMANDOS DO GESKTOP3D

Na tabela 5.2.5.1 são apresentados os comando utilizados e a maneira para acioná-los de acordo com o dispositivo utilizado, os gestos realizados são os mesmos no modo HandVu JNI e HandVu Socket.

Comando	Teclado	Mouse	Gestos
Movimentação do cursor no plano XY	Teclas ←, ↑, → e ↓	Movimentação do mouse no plano XY	Movimentação da mão no plano XY
Movimentação do cursor no eixo Z	Tecla A para cima e Z para baixo.	Tecla A para cima e Z para baixo.	Movimentação da mão para cima e para baixo
Seleção de arquivo e pilha no modo *desktop*	Tecla 1	Tecla 1	Lback
Seleção de mais de um arquivo ou pilha	Com um arquivo ou pilha selecionado, segurar a tecla CTRL e pressionar a tecla 1	Com um arquivo ou pilha selecionado, segurar a tecla CTRL e pressionar a tecla 1	Com um arquivo ou pilha selecionado, segurar a tecla CTRL e realizar Lback
Movimentar arquivo ou pilha	Pressionar e segurar a tecla 1 e movimentar o cursor	Pressionar e segurar a tecla 1 e movimentar o cursor	Realizar Lback e movimentar o cursor
Entrar no modo de movimentação de câmera	Pressionar a tecla SPACE	Pressionar a tecla SPACE	Realizar Victory
Sair do modo de movimentação de câmera	Pressionar a tecla SPACE	Pressionar a tecla SPACE	Realizar Open
Empilhar arquivos	Selecionar mais de um arquivo, segurar e pressionar a tecla 2 e pressionar a tecla A	Selecionar mais de um arquivo, segurar e pressionar a tecla 2 e pressionar a tecla A	Selecionar mais de um arquivo, fazer Closed e levantar a mão.

Desempilhar arquivos	Selecionar uma pilha, segurar e pressionar a tecla 2 e pressionar a tecla Z	Selecionar uma pilha, segurar e pressionar a tecla 2 e pressionar a tecla Z	Selecionar uma pilha, fazer Closed e baixar a mão
Ativar o modo de visualização de arquivos da pilha	Selecionar uma pilha e pressionar a tecla 3	Selecionar uma pilha e pressionar a tecla 3	Selecionar uma pilha e realizar LPalm
Selecionar arquivo no modo de visualização de arquivos da pilha	Pressionar a tecla ↓	Movimentar o *mouse* para baixo.	Movimentar mão para trás.
Sair do modo de visualização de arquivos da pilha	Pressionar a tecla 3	Pressionar a tecla 3	Realizar LPalm

Tabela 5.2.5.1 – Comandos do Gesktop3D

A figura 5.2.5.1 apresenta o Gesktop3D sendo utilizado através de gestos. A tela de captura do HandVu é mantida como referência para o usuário.

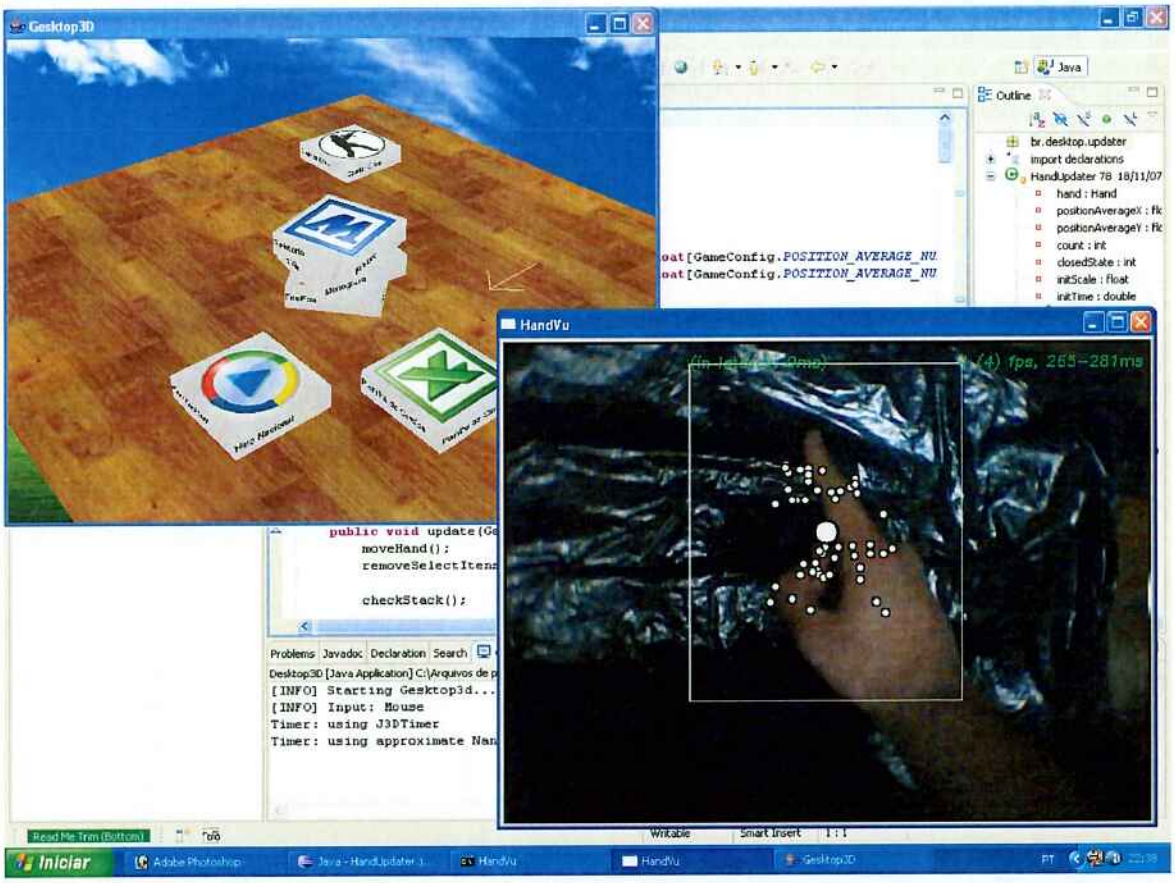


Figura 5.2.5.1 – Gesktop3D utilizado com gestos.

5.2.6 PRINCIPAIS PROBLEMAS DE DESENVOLVIMENTO

Além do problema encontrado no enJine citado posteriormente, houve outros problemas nas funções do desktop destacados a seguir:

- Diferença na intensidade dos dispositivos utilizados – Os três dispositivos utilizados apresentam os valores de intensidade distintos, o teclado apresenta o valor zero para a tecla não pressionada e um caso contrario. O mouse retorna os valores entre zero e a dimensão da tela, ou seja, para uma tela de 640 x 480 pixels, o valor da posição do mouse no eixo X é um valor real entre zero e 640 e no eixo Y entre zero e 480. Já o HandVu apresenta valores reais entre zero e um em ambos os eixos. Para resolver esse problema para todas as funções que utilizam posicionamento foram desenvolvidas duas versões para as mesmas, uma para o teclado e outra para o HandVu. E no caso do mouse há uma classe de conversor dos valores em escala para um valor real entre zero e um.
- Controle do cursor baseado no plano de coordenadas – Toda a movimentação do cursor é realizada a partir do plano de coordenadas da aplicação e não com relação à visão da câmera. Assim, caso o usuário modifique o posicionamento da câmera, os comandos de deslocamento do cursor poderão estar invertidos com relação à visão.
- Concorrência dos botões do mouse – Outro problema encontrado é a falta de comandos simultâneos do mouse, ou seja, não é possível clicar e segurar um botão do mouse e movimenta-lo ao mesmo tempo, por isso, não foi utilizado os botões do mouse, mas o teclado para os comandos do Gesktop3D.
- Direção de movimentação – Devido ao modo de utilização do mouse no enJine não é possível utilizar a direção do movimento, já que o Java retorna apenas a posição do cursor no instante de tempo verificado. Assim, para utilizar a direção do movimento é realizada uma cópia da

posição do cursor a cada deslocamento maior que dez pixels em relação à posição anterior. Com isso, a partir da posição gravada e da posição atual, faz se um cálculo para se obter o vetor de deslocamento do cursor.

- Colisão de arquivos quando esses são retirados da pilha – Quando um arquivo é retirado da pilha, esse retorna a sua posição de origem antes de estar na pilha, e, se há um outro objeto nesse mesmo espaço, ambos se sobrepõem. Nesse caso, para toda colisão entre arquivos, a aplicação tenta deslocar os arquivos em direção contrária até que não haja mais colisão.

5.3 INTEGRAÇÃO DO HANDVU AO ENJINE

A integração das duas ferramentas foi separada em 3 camadas.

- Camada de execução do HandVu;
- Porte do HandVu para Java;
- Integração da ferramenta com o enJine.

A figura a baixo ilustra as camadas da integração:

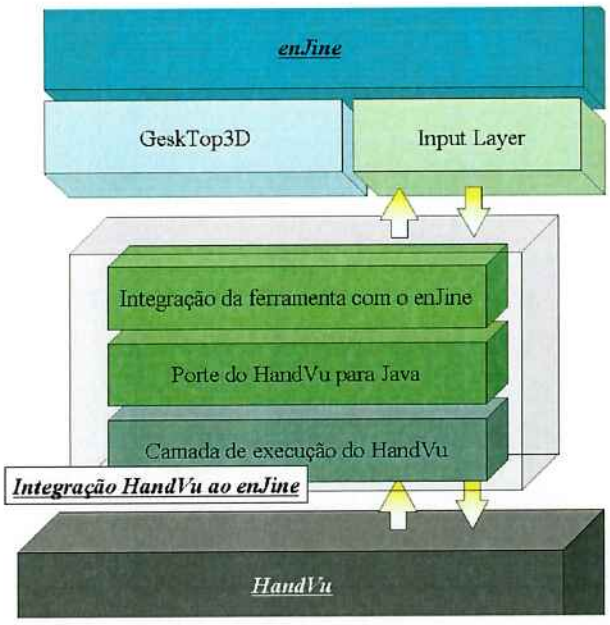


Figura 5.3.1 – Camadas

A camada de execução do HandVu, tem como principal característica a utilização da biblioteca do HandVu para o tracking e reconhecimento da postura da mão.

Para tal, foi implementaoa, na linguagem C++, um sistema que utiliza o HandVu para os reconhecimentos dos parâmetros como posição X e Y, escala e postura da mão a cada frame capturado pela câmera. Aqui foi criado uma função DirectShowBack, chamada pelo openCV, a cada frame, para tratar os frames capturados.

Nesta função o método hvProcessFrame(IplImage), utilizado pelo HandVu, é chamado para processar o frame e obter as informações necessárias. Aqui também é feita a conversão da imagem capturada pelo OpenCV(padrão BGR) para o padrão RGB utilizando-se o método cvCvtColor, do openCV.

Outra tarefa executada nesta função é o cálculo do tempo levado para o processamento entre frames, para tal são utilizados os métodos do openCV cvGetTickCount, que retorna o valor de ciclos do sistema, e o cvGetTickFrequency, que retorna a freqüência do clock. A divisão dos dois valores retorna o tempo decorrido em microssegundos, e a diferença entre o tempo obtido no processamento deste frame e do ultimo frame é o tempo utilizado para obter a informação de fps (frames per second).

Deve se tomar um cuidado para compilar esta camada em C++, pois ela deve ser compilada como uma DLL, para tal é necessário alterar um parâmetro do compilador. O APÊNDICE A mostra como compilar o HandVu e como compilar o Dll desta camada no Visual Studio 2003.

A camada de porte do HandVu para Java utiliza a biblioteca JNI para o porte do código da camada de execução do HandVu, em C++, para a linguagem Java. Nesta camada estão às chamadas natives para funções da camada inferior como:

- initCam: inicializa o processo de captura de dados;
- closeCam: termina a captura de dados;
- getX, getY, getPosture, getScale: obtém as informações capturadas da mão;

- getImageRGBData(byte[] imageArray): obtém os dados da imagem capturada e é colocada em imageArray no formato RGB;
- getLastMSPF: obtém a informação do tempo decorrido entre o processamento do penúltimo frame ao ultimo, em milissegundos.

O objeto que representa o HandVu em Java foi implementado com um singleton (GAMMA; VLISSES; JOHNSON; HELM, 1994), para que não possa ocorrer duas instancias simultâneas do HandVu utilizando a mesma câmera.

Como teste, foi criado um programa onde a rotação de um cubo, que possui como textura de suas 6 faces a imagem capturada pela câmera, é controlada pela posição da mão rastreada. A figura 5.3.2 mostra o aplicativo rodando.

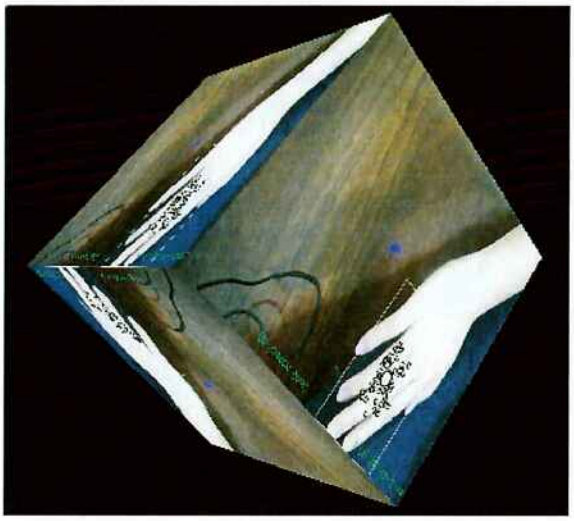


Figura 5.3.2 – Programa teste da camada de integração HandVu - Java

Na camada de integração com o enJine, foi criado um InputDevice, o HandVuDevice, para o HandVu, que contem InputSensors para a posição, postura e escala da mão. Este InputDevice criado também foi implementado como um singleton, pelo mesmo motivo que o objeto que representa o HandVu.

Quando o Device é instanciado ele inicializa o HandVu para a captura de dados, pausa por 10 segundos (tempo necessário para a inicialização da câmera), depois ele entra em um loop onde os dados dos sensores são atualizados e, antes de serem atualizados novamente o Device faz outra pausa pelo tempo obtido pela função getLastMSPF, para tentar ter o mesmo update rate que o frame rate do HandVu.

Outra abordagem para a integração do HandVu com o enJine foi a utilização do aplicativo do HandVu e Sockets. O aplicativo do HandVu quando executado, envia as informações capturadas para o localhost na porta 7045. O dado enviado para a porta é da seguinte forma:

1.2 tstamp id: t, r, "posture" (x, y) [s, a]

Onde:

- **1.2** é a versão do protocolo utilizado;
- **tstamp**: o tempo decorrido do frame atual desde o primeiro frame capturado;
- **id**: corresponde a qual objeto o evento corresponde, atualmente ele é fixo em 0;
- **t**: 1 se um objeto está sendo rastreado, caso contrário o valor é 0;
- **r**: 1 se alguma postura está sendo reconhecido, caso contrário o valor é 0;
- **"posture"**: a postura reconhecida;
- **(x,y)**: a posição da mão no formato (x,y);
- **[s,a]**: s corresponde a escala obtida. a é o valor de rotação, porem este último não está implementado na versão utilizada

Um exemplo de valor capturado utilizando o comando telnet é:

1.2 49671000 0: 1, 1, "closed" (0.703125, 0.494792) [8.491528, 0.000000]

Utilizando este aplicativo, foi criado o outro Device para o enJine, o **SocketHandVuDevice**, com os mesmos sensores que o **HandVuDevice**, porém utilizando a captura de dados por sockets.

Ambos os devices possuem vantagens e desvantagens como indicado na tabela 5.3.1.

	Vantagens	Desvantagens
HandVuDevice utilziando o JNI	• Maior controle do ambiente de execução; • Customização de métodos; • Recuperação pelo Java da imagem capturada.	• Inicialização com tempo de espera de 10 segundos; • Janela do HandVu não é fechada e possui problemas no controle deste; • Update rate do device não é totalmente igual ao frame rate do HandVu, uma vez que o tempo de espera sempre é o do frame anterior. • Alta taxa da falha de captura da postura.
SocketsHandVuDevice utilizando Sockets	• Update rate igual ao frame rate da captura de dados do HandVu; • Valor da postura obtido com maior freqüência. • Inicialização sem tempo de espera.	• Necessidade da inicialização do aplicativo do handVu antes da execução do enJine; • Janela das imagens capturadas diferente do aplicativo do enJine; • Impossibilidade da recuperação em Java da imagem capturada.

Tabela 5.3.1 – Vantagens e desvantagens de cada classe Device.

5.4 GESKTOP3D – INTEGRAÇÃO DO HANDVU AO DESKTOP 3D

Para a integração dos gestos, foram utilizados os dois devices criados, o HandVuDevice e SocketHandVuDevice. Uma vez que ambos possuem os mesmos InputSensors não é necessária a diferenciação dos dois devices no código. O único cuidado tomado é na inicialização, onde somente um dos devices é instanciado, uma vez que não é possível instanciar ambos os devices ao mesmo tempo. Isso ocorre, pois ambas instancias tentam acessar a mesma câmera e dessa formar uma delas não terá acesso a câmera, levando a ocorrência de um erro.

5.4.1 MOVIMENTAÇÃO DO CURSOR

Para a movimentação do cursor são utilizados os valores de posição da mão capturados pelo HandVu, eles são tratados de forma a variarem entre -0,5 e 0,5.

Depois eles são multiplicados por um fator de sensibilidade, isso é feito para que o usuário não necessite movimentar a mão por toda área capturada pela câmera. Assim o usuário pode percorrer todo o espaço do desktop com uma menor movimentação da mão, como é ilustrado na figura 5.4.4.1.

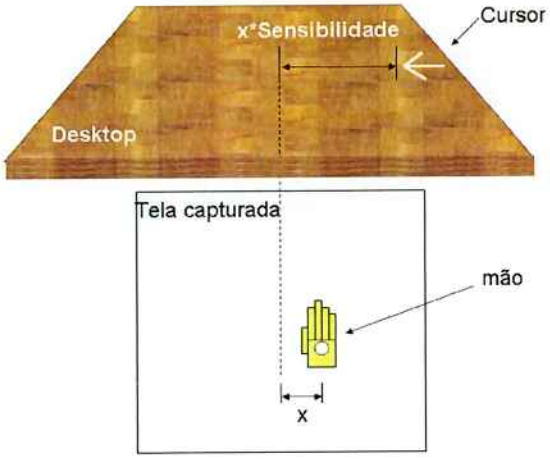


Figura 5.4.1.1 – Fator de sensibilidade

Na movimentação do cursor também é levado em conta um parâmetro do GameConfig, o USE_POSTURE_LOCK. Caso este parâmetro seja true o cursor é travado quando é detectada uma postura. Isso é implementado para que não ocorra de o usuário selecionar ou deselecionar um objeto indevidamente devido à oscilação da posição da mão detectada pelo HandVu.

Outra medida tomada para evitar a oscilação do valor da posição capturada pelo HandVu é o uso de uma média dos últimos n valores capturados, onde n é o valor contido em POSITION_AVERAGE_NUM no GameConfig. Com isso parte da oscilação é reduzida. O uso deste procedimento pode ser controlado alterando o valor de USE_POSITION_AVERAGE em GameConfig.

5.4.2 SELEÇÃO DE ARQUIVO

Para a seleção de arquivos é feita uma verificação de quando o cursor colide com um arquivo e se a postura da mão é igual a Lback, postura que ativa a seleção.

5.4.3 GESTO DE EMPILHAR/DESEMPILHAR

O gesto de empilhar e desempilhar é implementado utilizando-se a função checkStack, chamada na iteração de update do objeto. Aqui é verificado se uma postura Closed é detectada. Caso positivo um flag é alterado de forma a marcar que a postura Closed foi detectada e os valores de tempo e da escala da mão neste instante são armazenados.

Caso a postura Closed já foi detectada, ele verifica se ouve uma elevação da escala da mão suficiente (i.e. se a diferença da escala for maior que o valor STACK_SCALE_DIF de GameConfig), caso positivo ele altera o flag para que se empilhe os arquivos selecionados. O gesto de desempilhar é implementado de forma análoga.

Caso passe um tempo maior que STACK_TIME_OUT sem que ocorra variação significativa na escala da mão, o flag é anulado e inicia-se a detecção da postura Closed novamente.

O diagrama de estado da figura 5.4.3.1 ilustra o fluxo de eventos.

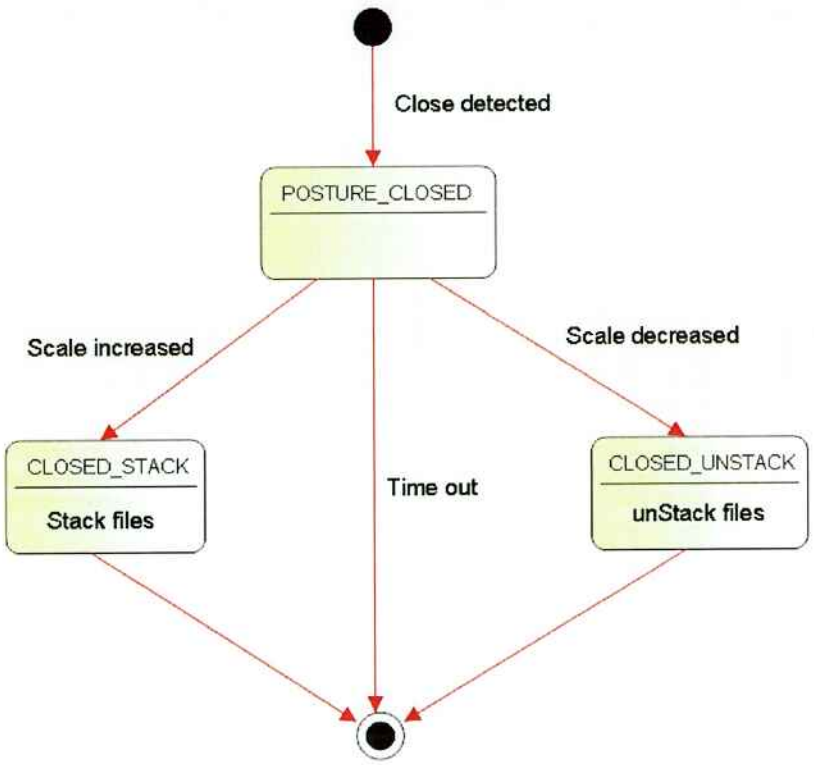


Figura 5.4.3.1 – Diagrama de estados do gesto de empilhar

5.4.4 SELEÇÃO E MANIPULAÇÃO DE PILHAS

A função processStack foi alterada de forma a verificar se a postura capturada é Lback para a seleção e se ela é LPalm para alterar do modo do desktop para o modo de manipulação da pilha.

5.4.5 ALTERAÇÃO PARA OS MODOS DE CÂMERA - DESKTOP

No update do GameState foi acrescentada uma verificação das posturas Victory e Open. Caso a postura Victory é detectada o modo do desktop é

alterado para o modo de câmera, e caso a postura Open for detectada, o modo é alterado para o desktop.

5.4.6 MANIPULAÇÃO DE CÂMERA

Uma vez no modo de câmera, os inputs fornecidos pelo HandVu são utilizados para a manipulação da câmera. Caso o usuário altere a posição no eixo X da mão, a câmera se movimenta para os lados seguindo uma trajetória circular em relação ao centro do desktop. A movimentação no eixo Y faz com que a câmera de um zoom-in ou um zoom-out. Ambas as movimentações são ilustradas nas figuras 5.4.6.1 e 5.4.6.2.

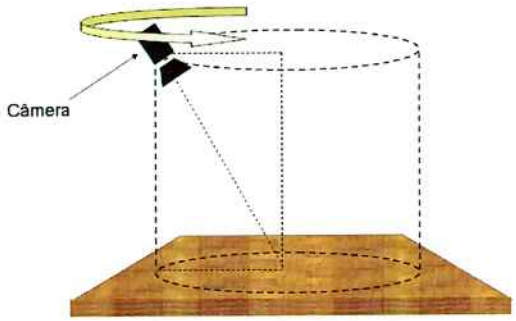


Figura 5.4.6.1 – Rotação da câmera

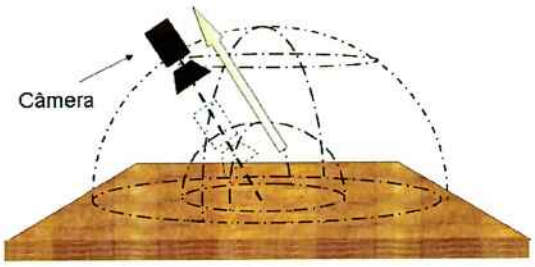


Figura 5.4.6.2 – Zoom-in, zoom-out

Acionando a postura Lpalm, o modo é alterado de forma que a posição Y da mão é utilizada para controlar a altura que a câmera se encontra, como ilustrada na figura Z. Para voltar ao outro modo de controle basta o usuário acionar a postura de Lback.

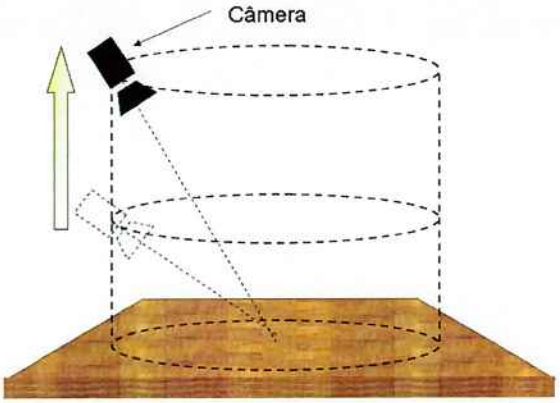


Figura 5.4.6.3 – Alteração da altura da câmera

5.4.7 MANIPULAÇÃO NO MODO DE VISUALIZAÇÃO DE PILHA

Aqui os movimentos da mão são mapeados de forma semelhante ao mouse. Onde a movimentação no eixo X da câmera altera o arquivo focado, o movimento para baixo, no eixo Y, seleciona o arquivo e o movimento para cima executa o arquivo.

6. TESTES

Para poder verificar algumas viabilidades do sistema e para podermos concluir como qualidade da interação por gestos no sistema desenvolvido, foram planejados e realizados alguns testes, que serão melhor detalhados abaixo.

6.1 HANDVU – DETECÇÃO DE POSTURAS

Para verificar o uso das posturas que o HandVu reconhece, foram realizados testes, com quatro pessoas, para definir o tempo de resposta de reconhecimento dessas posturas. Com o resultado desses testes foi possível determinar quais as melhores posturas para serem utilizadas.

Os testes foram realizados com o seguinte hardware: Processador Intel Core 2 Duo T5500 1.66 GHz., memória RAM 2GB, Placa de Vídeo Mobile Intel 945GM Express Chipset, Câmera Logitech QuickCam Communicate STX.

Todos os testes foram realizados no mesmo ambiente e em horários próximos e com as mesmas condições de iluminação.

6.2 TESTES FORMAIS DE USABILIDADE

Após a conclusão do desenvolvimento, planejou-se um teste de usabilidade para verificar as vantagens e desvantagens do uso do Desktop3D desenvolvido (em comparação com Desktops 2D tradicionais) e a adequação do uso de gestos como forma de interação do sistema.

Para a realização desse teste temos alguns requisitos (LEWIS; RIEMAN, 1994) que são: utilização de um laboratório especial para testes que possua duas salas, sendo que uma delas será utilizada para os testes e a outra para

observação. Essas salas devem estar separadas por um espelho, possibilitando a observação do usuário que está realizando o teste pelo observador que se encontrará na outra sala. Essa separação é utilizada para que o usuário que realiza o teste não seja influenciado por reações do observador. Junto com o usuário que realiza o teste deve-se ter um entrevistador para que o usuário não se sinta isolado e para encorajá-lo a realizar comentários em voz alta. Também é necessário filmar a realização do teste e armazenar essa filmagem. Todas as ações realizadas pelo usuário no sistema devem também ser armazenadas em logs.

Também é necessária a definição de algumas características que o teste deve possuir (MAYHEW, 1999). Primeiramente deve-se definir o tipo do teste e para esse projeto optou-se pelo teste de comparação, no qual se determina tarefas a serem realizadas. Então essas tarefas são realizadas em sistemas semelhantes possibilitando uma avaliação comparativamente a satisfação e produtividade do usuário nos sistemas testados. Nesse projeto os sistemas que serão comparados nesse teste serão: Gesktop3D com teclado, com teclado e mouse, com gestos e teclado e finalmente apenas com gestos.

Após a definição do tipo de teste, defini-se qual será o público alvo para realização desses testes. Para esse teste teremos o seguinte público alvo: usuários com idade entre 20 e 30 anos, podendo ser tanto do sexo masculino como do feminino. Um ou mais anos de experiência com a utilização de computadores e também um treinamento mínimo para utilização do Gesktop, sendo que esse treinamento poderá ser dando antes da realização do teste e dura 15 minutos. Nesse treinamento o apresenta-se todos os comandos e funcionalidades do Gesktop, e após essa apresentação o usuário tem um tempo para se familiarizar com o aplicativo podendo tirar dúvidas sobre o funcionamento com o entrevistador que o acompanhará nessa parte do processo.

Então se define um cenário inicial para os testes. Nesse projeto, o cenário inicial possui: 10 arquivos de música pertencentes a 3 bandas diferentes (x1, x2, x3, x4, y1, y2, y3, z1, z2, z3), 3 arquivos com letra de uma das 10 musicas em cada um deles (x1, y2, z3) e outros seis arquivos aleatórios que não podem ser nem de música nem de letras de músicas.

Tendo o cenário inicial definido, as tarefas a serem realizadas são definidas. As tarefas que devem ser cumpridas nesse projeto são: agrupar todos os arquivos de música, separando-os dos demais arquivos; separar desses arquivos de música que foram agrupados o arquivo x1.música; agrupar em um local diferente todos os arquivos de música que começam com y, separando-os dos demais; agrupar os arquivos x1.musica e x1.letra; executar o arquivo x1.musica e abrir o arquivo x1.letra; executar todos os arquivos de música que tem o nome iniciado por y; reagrupar os arquivos de música, juntando no mesmo grupo todos os que começam com a mesma letra.

O próximo passo é a definição das métricas que serão avaliadas. Baseando-se na ISO 9241 (ISO9241, 1998) e na ISO 9126 (ISO9126, 2001) os atributos escolhidos para serem avaliados nesse projeto são: Operabilidade, medida através da quantidade de erros e do tempo para realização de cada tarefa. Apreendabilidade, medida através da quantidade de erros, do número de passos para realização de uma tarefa e pela quantidade total de tarefas realizadas. Satisfação, medida através da avaliação do questionário respondido pelo usuário ao final do teste.

Para realizar essas medições no sistema, definiu-se que: erro será toda ação realizada pelo usuário que não leva a atingir o resultado esperado; a contagem do número de passos será feita levando em consideração a realização das seguintes ações: selecionar e deselecionar objeto, troca da postura da mão, necessidade de reconhecer a mão novamente, empilhar e desempilhar arquivos, arrastar arquivos ou pilhas, alterar modo de visualização, mover a câmera mudando o ângulo de visão, mudança de arquivo no modo de visualização de uma pilha, execução de um arquivo.

Para finalizar a definição do teste, definimos quais serão os questionários pré e pós-teste que devem ser respondidos pelos usuários que irão realizar o teste, a fim de completar os dados já mencionados.

O questionário pré-teste contém perguntas para verificar se o usuário realmente se enquadra no público alvo, sendo que esse questionário será o seguinte:

Questionário Pré-teste	ID:

Preencha o questionário abaixo:
Idade:

Sexo:

Tempo de utilização de computador:

Realizou treinamento:

Figura 6.2.1 – Questionário Pré-teste

O questionário pós-teste contém as informações para verificação da satisfação do usuário com o sistema e será como o mostrado abaixo:

Questionário Pós-teste	ID:

Para cada linha da tabela abaixo escolha apenas uma coluna marcando-a com um X:

5- Muito Bom; 4- Bom; 3- Indiferente; 2- Ruim; 1- Muito Ruim

	5	4	3	2	1
Facilidade de uso					
Facilidade de navegação					
Utilidade					
Facilidade de aprendizado					
Facilidade de se recuperar de erros					
Avaliação Geral					

Comentários:

Figura 6.2.2 – Questionário Pós-teste

6.3 TESTES INFORMAIS DE USABILIDADE

Devido à falta de tempo para realização dos testes formais, optou-se pela realização desses testes de forma mais informal, seguindo somente parte das recomendações listadas em 6.2 e com um grupo de usuários menor. Todas as tarefas indicadas foram realizadas, as métricas estabelecidas foram seguidas e medidas e os questionários foram respondidos.

6.4 TESTES COM O HANDVU APÓS INTEGRAÇÃO

Com o termino da implementação do Gesktop3D foi possível constatar que o HandVu possui alguns problemas, tais como:

- Instabilidade da posição da mão durante a detecção de uma postura.
- Alteração da posição da mão entre a troca de posturas.

Um dos problemas é a instabilidade dos valores de posições X e Y da mão capturados pelo HandVu. Quando se exerce uma postura os valores não se estabilizam se alterando consideravelmente, mesmo quando mantêm a mão imóvel, diferentemente dos valores de quando não é exercida uma postura. Outro problema está na variação de valor da posição durante a alteração de posturas. Devido ao fato do HandVu definir a posição da mão como sendo o centro da imagem da mão capturada, alterações consideráveis do formato da mão faz com que este centro se altere, e assim alterando a posição da mão do usuário, mesmo ele não tendo movido a mão.

A figura 6.4.3 mostra a alteração de quando não se exerce nenhuma postura para a postura "Closed". Pode se notar a alteração da posição do circulo branco, que representa a posição da mão capturada pelo HandVu.

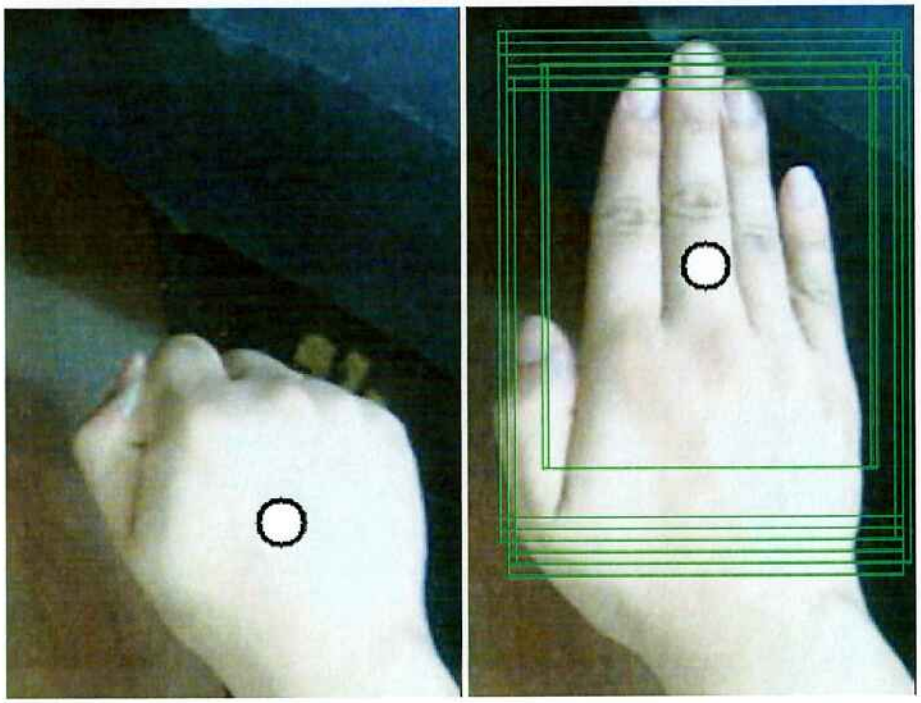


Figura 6.4.1 – Alteração da posição da mão devido à alteração da postura

Para testar estas características foram feitos testes para capturar os valores de posição em cada situação descrita. Os valores obtidos são discutidos nos resultados.

Durante o desenvolvimento foram coletados outros dados do da integração alem do HandVu em si. Foram constatados alguns problemas como:

- Impossibilidade do controle da janela de captação de imagens pelo HandVu
- Aumento na taxa de erros na detecção de posturas

Com o porte do HandVu, e a captura da imagem da câmera no Java, não há mais a necessidade da utilização da janela criada pelo HandVu a visualização da imagem de câmera. Porem não foi possível retirar ou mesmo ocultar esta janela. Outro problema ocorre quando o foco da janela é retirado uma vez, pois após isso não é mais possível visualizar as imagens capturadas. Também não é possível mover e redimensionar a janela.

Outro problema do porte do HandVu foi que a taxa de acerto na detecção de uma postura cai drasticamente. A figura abaixo mostra a taxa de acerto do aplicativo do HandVu original e a taxa de acerto do porte do JNI, para este teste a postura foi mantida constantemente e foi verificado a taxa de acerto.

Para validar esta taxa de acerto foram executados testes mantendo-se algumas posturas e verificando qual é a taxa de acerto de cada frame processado.

6.5 TESTES DE DESEMPENHO

Foram, também, realizados testes em máquinas diferentes, uma com maior poder computacional para verificar o desempenho do sistema.

As configurações Das máquinas são:

Maquina 1:

- Processador: Core 2 Duo T5500 1.66 GHz
- Memória: 2GB RAM
- Placa gráfica: Onborad, 128 MB de RAM.

Maquina 2:

- Processador: Core 2 Quad Q6600 2.4GHz
- Memória: 3.25GB RAM
- Placa gráfica: Geforce 7300

Os testes de desempenho são para verificar o numero de frames processados em cada ambiente.

7. CONSIDERAÇÕES FINAIS

Nesse capítulo serão apresentados os resultados de todos os testes descritos no capítulo 6. Também serão apresentadas as conclusões feitas com a realização desse projeto.

7.1 RESULTADOS

Nessa sessão serão avaliados os resultados obtidos com a realização dos testes descritos no capítulo 6.

7.1.1 DETECÇÃO DAS POSTURAS NO HANDVU

Os resultados são apresentados a seguir, sendo que esse teste foi realizado na maquina 1, citada no item 6.5 e a detecção da mão, é a medida do tempo necessário para o HandVu iniciar o rastreamento da mão de maneira contínua.

		Usuários				Média Geral
		Usuário 1	Usuário 2	Usuário 3	Usuário 4	
Postura detectada	Detecção da mão	3,3s	2,5s	3,5s	0,9s	3,4s
	Lback	2,6s	2,3s	3,6s	1,6s	2,5s
	Lpalm	2,6s	2s	6s	1,3s	2,9s
	Close	6s	6,3s	3,3s	2,3s	4,5s
	Open	1,5s	1,6s	NE	2,3s	1,8s
	Victory	1,5s	2s	3,3s	1,3s	2s
	Sidepoint	4,6s*	ND	ND	18,6s	18,6s

* - Não manteve a reconhecimento da postura de forma constante;
ND - Postura não detectada; NE - Teste não realizado.

Tabela 7.1.1.1 – Tempos obtidos na detecção da mão e de posturas

Como pode-se observar nos resultados desse teste, o HandVu apresenta dificuldade para reconhecer a postura sidepoint. Devido a isso, essa postura não será utilizada.

7.1.2 TESTES INFORMAIS DE USABILIDADE

Os usuários que realizaram o teste realizaram todas as tarefas solicitadas e essas tarefas foram:

1. Agrupar todos os arquivos de música, separando-os dos demais arquivos;
2. Separar desses arquivos de música que foram agrupados o arquivo x1.musica;
3. Agrupar em um local diferente todos os arquivos de música que começam com y, separando-os dos demais;
4. Agrupar os arquivos x1.musica e x1.letra;
5. Executar o arquivo x1.musica e abrir o arquivo x1.letra;
6. Executar todos os arquivos de música que tem o nome iniciado por y;
7. Reagrupar os arquivos de música, juntando no mesmo grupo todos os que começam com a mesma letra.

Os resultados coletados na realização dessas tarefas podem ser vistos no APÊNDICE B.

Após a realização das tarefas, esses usuários responderam o questionário pós-teste. Os resultados podem ser vistos no APÊNDICE B.

Analisando os resultados podemos perceber que na maioria das tarefas e para a maioria dos usuários, utilizando o HandVu, tanto o número de erros, como o tempo para realização das tarefas, como o número de passos são geralmente maiores. Analisando o questionário de satisfação, podemos perceber que em todos os quesitos analisados o HandVu tem uma pequena desvantagem sobre os outros métodos de interação. Temos também os comentários realizados pelos usuários durante a realização dos testes que nos

levam a perceber que eles ficaram muito satisfeitos em utilizar o aplicativo com os gestos.

7.1.3 TESTES COM O HANDVU APÓS INTEGRAÇÃO

Os gráficos abaixo mostram os valores de posições das duas situações comentadas nos testes do HandVu, testes realizados para verificar os valores de posição.

Como era esperado, pode-se ver no gráfico 7.1.3.1 que quando o HandVu não está detectando uma postura os valores da posição da mão praticamente não se alteram.

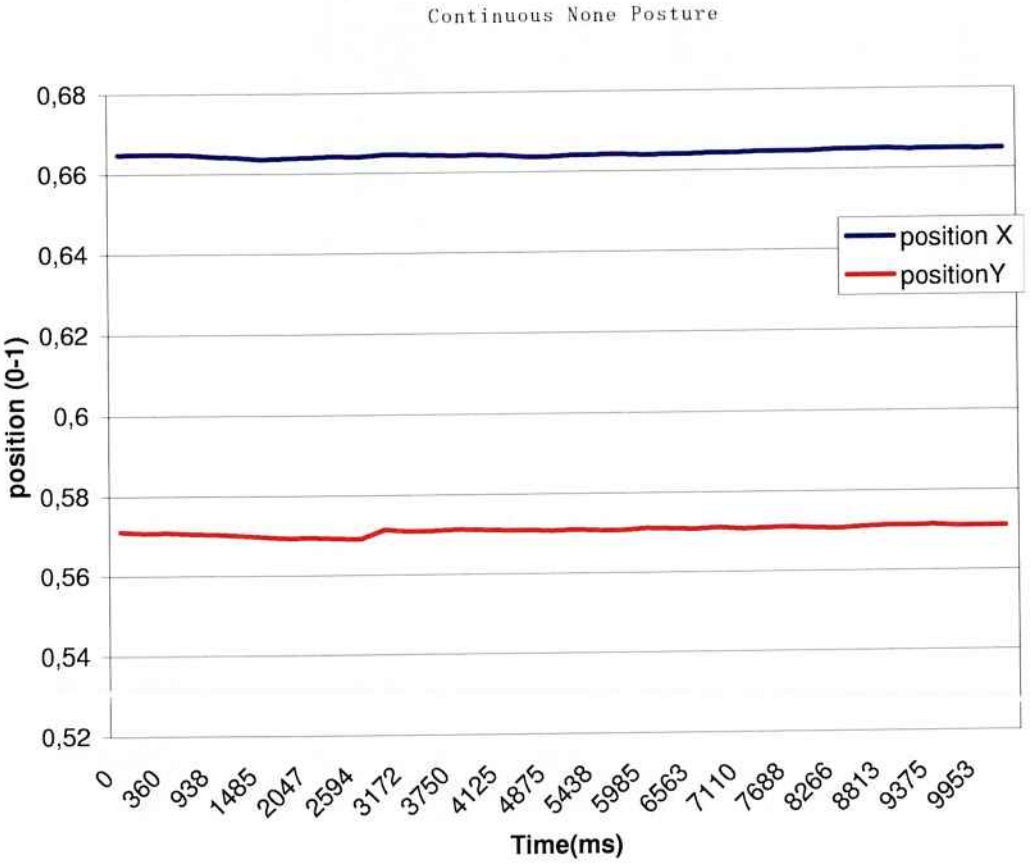


Gráfico 7.1.3.1 – Posições XY da mão sem postura

No gráfico 7.1.3.2 pode-se notar que quando uma postura é detectada, mesmo mantendo a mão sem movimento, os valores de posição se alteram não se estabilizando.

Esta instabilidade prejudica a navegação no Gesktop3D. Pois quando o usuário está movendo o cursor pela tela, este pode ir para um local não desejado e dessa forma o usuário pode perder tempo na realização de suas tarefas, ou mesmo perder a credibilidade no sistema, pois esse não responde a seus comandos.

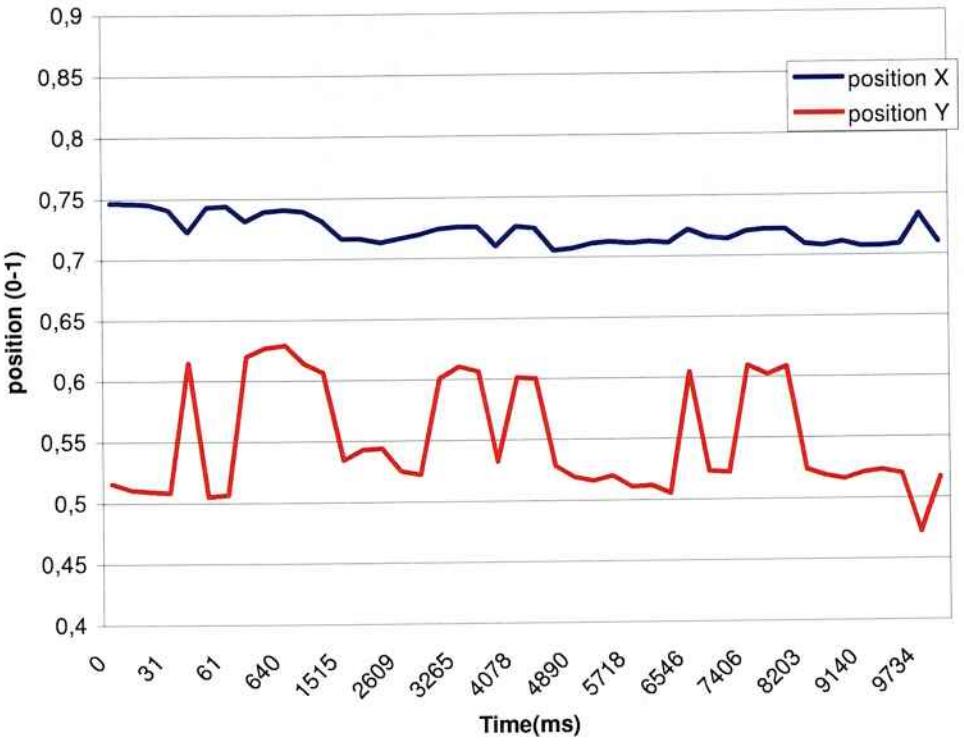


Gráfico 7.1.3.2 – Posições XY com a postura Closed

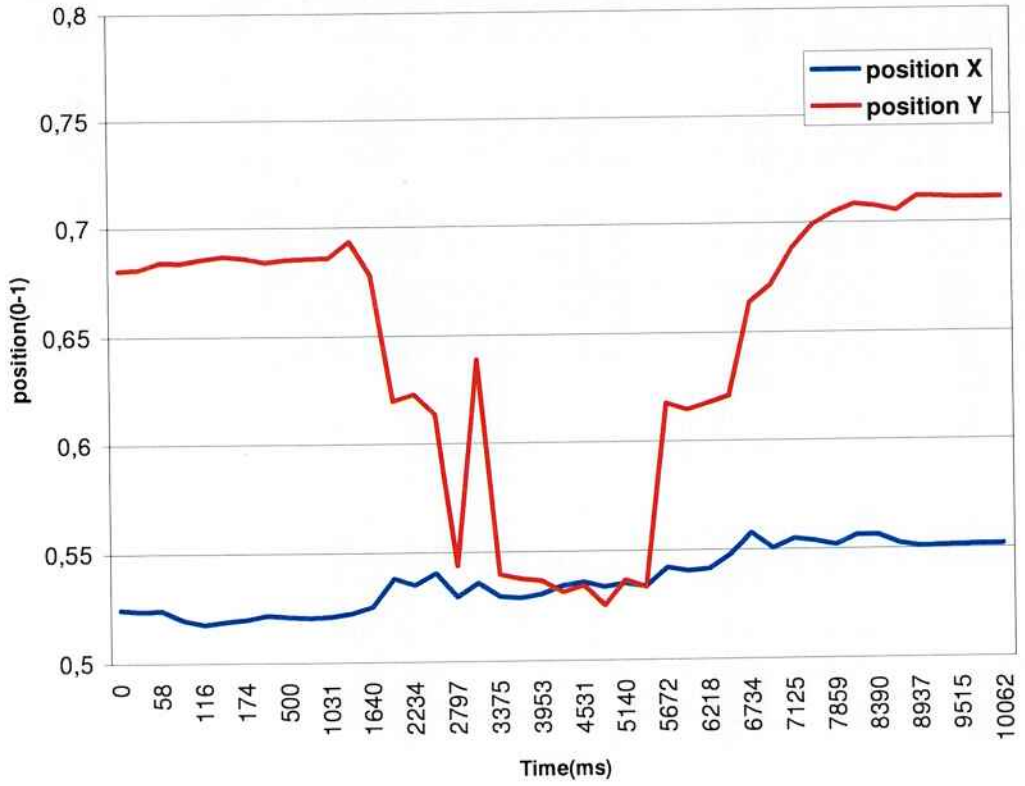


Gráfico 7.1.3.3 –Posições XY durante a alteração de postura

Outro resultado obtido é em relação à alteração do valor de posição quando se altera a postura da mão. Acima pode ser visto o gráfico com os valores capturados nessa situação. É importante notar uma considerável alteração da posição Y.

No uso de Gesktop3D, este é um dos principais problemas. Pois, quando o usuário tenta selecionar um arquivo, alterando a postura, o cursor irá se movimentar, fazendo com que este se desloque do local desejado pele usuário, isso faz com que o usuário selecione algo não desejado, ou mesmo "deselecione" todos os itens anteriormente selecionados.

Também foi realizado o teste da taxa de acerto dos valores de postura capturados com o porte de Sockets e o que utiliza o JNI. O Gráfico 7.1.3.4 mostra a diferença entre as taxas de acerto.

	Sistema Operacional	Threads
closed	96,4	54,2
open	92,5	28,0
lback	97,3	24,1
lpalm	97,4	39,4
victory	96,1	39,0

Tabela 7.1.3.1 – Comparação de taxa de acertos

Pode-se verificar que a taxa de acerto do porte é muito menor do que a aplicativo original.

7.1.4 TESTES DE DESEMPENHO

Dos testes de desempenho obtiveram-se os seguintes dados. A tabela seguinte mostra os valores de FPS (frames per second).

	Maquina 1	Maquina 2
closed	7,3	11
open	5,9	8
lback	8,6	11,2
lpalm	8,4	11,5
victory	7,6	10,3

Tabela 7.1.4.1 – Número de frames processados

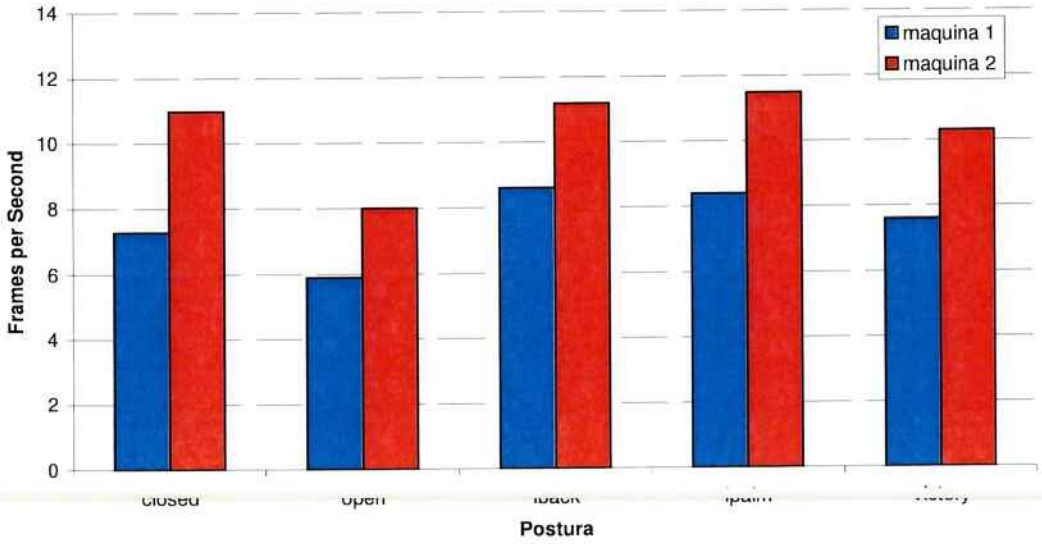


Gráfico 7.1.4.2 – Frames processados para cada postura por maquina

Nota-se que o número de frames processados em geral pela maquina 2 é maior. Este fato faz com que a resposta do sistema seja melhor, pois o número de frames processados em um determinado período de tempo é maior, aumentando as chances de se detectar a postura correta, alem de que este aumento também faz com que o sistema tenha maior fluidez nos movimentos.

7.2 CONCLUSÕES

Nesta monografia discutiu-se o uso de gestos como interface em ambientes tridimensionais. Para isso, foi desenvolvido um ambiente virtual de desktop 3D controlado através de gestos. Essa aplicação utiliza como ferramentas principais um engine didático de jogos, o enJine e o HandVu, aplicação que reconhece gestos e faz o rastreamento dos movimentos da mão do usuário. De modo a atingir os objetivo, ambas as ferramentas foram estudadas a fim de integrá-las usando a JNI, possibilitando o uso do HandVu como uma biblioteca no Java.

Na implementação do Desktop, há um cuidado especial quanto à usabilidade do Desktop, para isso foram utilizadas as oito regras de ouro de usabilidade de Shneiderman (SHNEIDERMAN, 2004), bem como as recomendações de Bowman et al. (BOWMAN, 2004), de modo a melhorar a qualidade de uso do Desktop.

Também foram realizados testes com diversos usuários que seguiram a metodologia proposta por Bowman et al. (BOWMAN, 2004) e pela ISO (ISO, 1998) (ISO, 2001). Esses testes procuraram verificar a eficiência do uso de gestos em interfaces 3D e a satisfação do usuário.

Baseado nos resultados encontrados, foi possível verificar que o uso de gestos possui grande potencial para a interatividade em ambientes 3D, devido ao fato de que, diferente dos dispositivos usuais atuais como o teclado e mouse, o uso de gestos permite uma movimentação no espaço mais natural. Contudo, devido à dificuldade de reconhecimento da postura em tempo real pelo HandVu,

principalmente em plataformas de hardware com menor capacidade de processamento, recomenda-se utilizar o teclado como auxilio à interação por gestos, de modo que o HandVu foi utilizado para movimentação do cursor e o teclado para realizar as ações.

Nessa linha, o HandVu pode ser utilizado para o rastreamento da mão sem reconhecer as posturas, por exemplo substituindo o touchpad de notebooks em locais que não é possível utilizar um mouse já que o uso do touchpad não é tão confortável quanto o uso de um mouse. Esta abordagem é utilizado no Gesktop3D no modo gestos mais teclado.

Também foi desenvolvido um framework, formado pela integração do HandVu e do enJine, que pode ser utilizado para o desenvolvimento de outras aplicações 3D com interação por gestos, trazendo diversas outras possibilidades de futuros trabalhos. Por exemplo, aplicações de apresentações ou teleconferência que os usuários mesmo movimentando-se no palco podem avançar na apresentação sem a necessidade de dispositivos adicionais.

Esta integração também pode ser utilizado em sistemas que utilizam gestos que não dependem da precisão do movimento, como controle binário, por exemplo um gesto para ligar e desligar as iluminações da casa.

Ainda há a possibilidade de uso em aplicações para usuários inexperientes. Uma vez que os gestos são um meio de interação natural para o usuário, o meio de interatividade torna-se mais interessante, tornando a curva de aprendizagem do sistema menor.

O projeto tem ainda como possíveis trabalhos futuros a projeção da aplicação em uma mesa, utilizando Realidade Aumentada Espacial (RASKAR, 2005) e a infra-estrutura do Robot ARena (CALIFE et al., 2007), de modo a trazer as funcionalidades do desktop virtual para a realidade e tornar o ambiente mais natural para o usuário. Assim, o controle do desktop virtual ocorre pela própria mão do usuário, aumentando sua imersão. Essa idéia é semelhante ao MS Surface (SURFACE, 2007).

Com relação à diferença da taxa de acerto do reconhecimento de postura entre a integração utilizando Sockets e a integração utilizando JNI foram levantadas duas hipóteses: concorrência do processamento entre Threads e o assincronismo entre os processos. Na primeira, supõe-se que a maquina virtual

do Java tenta balancear os processos, enquanto que o sistema operacional prioriza os processos com maior carga, assim a utilização JNI não é tão eficiente quanto a utilização de Sockts. Na segunda, supõe-se que ocorre uma assincronia entre os processos, HandVu e aplicação, desse modo pode ocorrer captura de dados desatualizados, acarretando a alta taxa de erros. Assim, torna-se um trabalho futuro a análise desse problema e a correção para melhorar o desempenho da utilização do HandVu com JNI.

Outro problema encontrado é a falta de um ponto fixo de referência no rastreamento da localização da mão no HandVu, devido a utilização do centro da área da imagem capturada da mão como posição da mão. Assim, ao mudar a postura, sem movimentar a mão, a posição capturada é deslocada, prejudicando o uso de mudanças de posturas em aplicativos que necessitam de precisão da posição. Uma possível solução seria utilizar um ponto fixo no pulso do usuário como referência para a posição da mão.

Enfim, apesar das dificuldades do uso do HandVu, de modo geral, notou-se que o uso de gestos torna a interface mais interessante de se utilizar, baseado nas respostas subjetivas de cada usuário, que indica um grande potencial do uso de gestos como meio interativo.

8. REFERÊNCIAS

AZUMA, R. A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, v .6, n.4, August 1997, 355-385 p.

BERNARDES, J.L.; NAKAMURA, R.; CALIFE, D.; TOKUNAGA, D.; TORI, R. Integrating the Wii controller with enJine: 3D interfaces extending the frontiers of a didactic game engine, 2007. 10 p.

BOWMAN, D. et al., "3D User Interfaces: Theory and Practice", Addison Wesley, 2004

CALIFE, D.; TOMOYOSE, A.; SPINOLA, D.; BERNARDES, J.L.; TORI, R. Robot ARena: Infrastructure for Applications Involving Spatial Augmented Reality and Robots. In: Proceedings of the IX Symposium on Virtual and Augmented Reality, 2007.

ENJINE. Incubadora Virtual: enJine: Engine para Jogos Online em Java. Disponível em: http://incubadora.fapesp.br/projects/enjine/. Acesso em: 10 abr. 2007.

EYETOY. EyeToy. Disponível em: http://www.eyetoy.com/. Acesso em: 10 abr. 2007.

FOWLER, M. "UML Essencial", Editora Bookman, 2005.

GAMMA, E.; VLISSIDES, J.; JOHNSON, R.; HELM, R. "Design Patterns Elements Of Reusable Object-Oriented Software", Addison Wesley, 1994

HANDVU. HandVu postures. Disponível em: http://www.movesinstitute.org/~kolsch/HandVu/doc/HandVu_Postures.pdf. Acesso em: 10 abr. 2007.

IPHONE. iPhone Apple. Disponível em: http://www.apple.com/iphone/. Acesso em: 10 abr. 2007.

IPOD. iPod Apple. Disponível em: http://www.apple.com/ipod/ipod.html. Acesso em: 10 abr. 2007.

JACOBSON, I., BOOCH, G., RUMBAUGH, J. The Unified Software Development Process. Addison-Wesley Object Technology Series, 1998

JAVA3D. Java 3D API. Disponível em: https://java3d.dev.java.net/. Acesso em: 22 nov. 2007.

KIRNER, C.; PINHO, M.S. Introdução à Realidade Virtual. Livro do Mini-curso, 1º Workshop de Realidade Virtual.São Carlos, SP. Nov. 1997.

KOLSCH, M. Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments, 2004. 208 p.

KOLSCH, M.; TURK, M. Analysis of Rotational Robustness of Hand Detection with a Viola-Jones Detector, 2004. 4 p.

LEWIS, C.; RIEMAN, J. Task-centered user interface design: a practical introduction, 1994.

MAYHEW, D.J. The usability engineering lifecycle: a practitioner's handbbok for user interface design. San Francisco: Morgan Kaufmann, 1999.

MILGRAM, P.; TAKEMURA, H.; UTSUMI, A.; KISHINO, F. Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. Telemanipulator and Telepresence Technologies, SPIE, v.2351, out. 1994, 282-292 p.

NAKAMURA, R.; BERNARDES, J.L.; TORI, R. enJine: Architecture and Application of an Open-Source Didactic Game Engine. In: Digital Proceedings of the Brazilian Symposium on Games and Digital Entertainment 2006, 2006. Disponível em: http://www.cin.ufpe.br/sbgames/proceedings/index.htm. Acesso em: 20 ago. 2007.

ORGANIZAÇÃO NACIONAL PARA PADRONIZAÇÃO. ISO9126: Software Engineering – Product Quality – elaboração 2001

ORGANIZAÇÃO NACIONAL PARA PADRONIZAÇÃO. ISO9241: Usability Standard – elaboração 1998

PAVLOVIC, V. I.; SHARMA, R.; HUANG, T. S. Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review, 1997. in IEEE Transaction on Pattern Analysis and Machine Intelligence, v. 19, n. 7, jul. 1997. 677-695 p.

RASKAR, R.; BIMBER, O. Spatial Augmented Reality: Merging Real an Virtual Worlds, A. K. Peters, 2005.

SHNEIDERMAN, B.; PLAISANT, C. Designing the User Interface, Addison Wesley, 4º edição.

SMITH, J. A Comparision of RUP and XP. Rational Software, 2002. (White Paper)

SURFACE. Microsoft Surface. Disponível em: http://www.microsoft.com/surface/. Acesso em: 18 nov. 2007.

TORI, R.; BERNARDES, J. AND NAKAMURA, R. Teaching Introductory Computer Graphics Using Java 3D, Games and Customized Software: a Brazilian Experience. In: Digital proceedings of the 34th International Conference and Exhibition on Computer Graphics and Interactive Techniques, Educators Program. ACM SIGGRAPH, 2006.

TORI, R.Negócios no Second Life – Será que vale a pena?. Revista Info Exame. Jul. 2007.

TSUDA, F.; HOKAMA, P.; RODRIGUES, T.; BERNARDES, J. Integration of jARToolKit and enJine: extending with AR the potential use of a didactic game engine. In: Proceedings of the IX Simposium on Virtual and Augmented Reality. SBC, 2007. 51-59 p.

VIOLA, P.; JONES, M. Fast Multi-view Face Detection, Technical Report TR2003-96, MERL, jul. 2003.

WII – The Global Wii Experience Website in English. Disponível em http://us.wii.com/. Acesso em 10 abr. 2007.

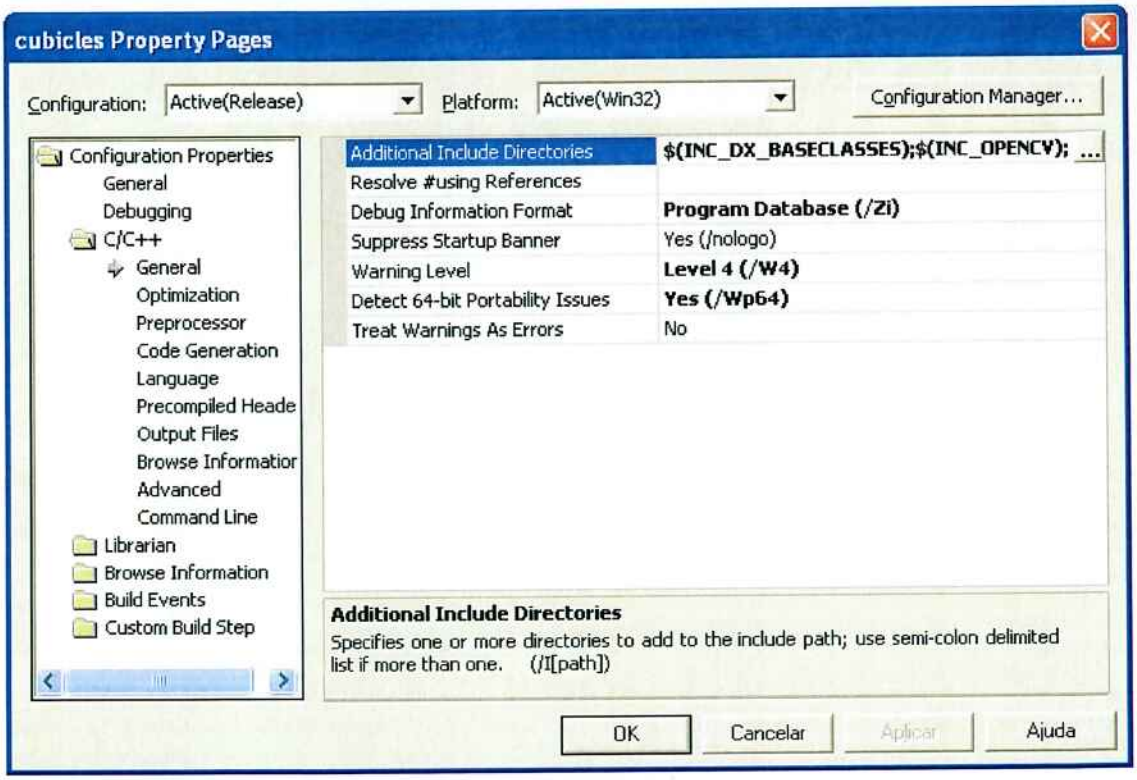
APÊNDICE A – COMPILAÇÃO DO HANDVU NO VISUAL STUDIO 2003

Passos necessários para compilar o HandVu no Visual Studio 2003

Pré-requisitos:
OpenCv beta 5 instalado
DirectX 9 SDK
Dev-C++

Antes de compilar o hv_cvCam precisamos compilar o projeto cubicles e HandVu para obter as libs: cubicles.lib e HandVu ou cubiclesd.lib e HandVud.lib se forem rodar o HandVu no modo debug (não consegui fazer isso funcionar)

Para compilar precisamos indicar ao Studio onde encontrar os headers e libs dependentes. Para isso, em Project -> (nome do projeto) Properties. Na nova janela ir na aba Configuration Properties -> C/C++ -> General. Em Additional Include Directories, há as dependências de cada projeto, pode-se colocar essas dependências nas variáveis de ambiente do Windows ou colocar o diretório dessas headers diretamente no Studio.



As variáveis e os caminhos das dependências estão listadas abaixo (considerando que o OpenCv foi instalado no diretório default e o direct X no diretório DXDSK):

Variavel	Caminho
$(INC_DX_BASECLASSES)	C:\DXSDK\Samples\C++\DirectShow\BaseClasses
$(INC_OPENCV)	C:\Arquivos de programas\OpenCV\cv\include

$(INC_OPENCV_CXCORE)	C:\Arquivos de programas\OpenCV\cxcore\include
$(INC_OPENCV_HIGHGUI)	C:\Arquivos de programas\OpenCV\otherlibs\highgui
$(INC_OPENCV_AUX)	C:\Arquivos de programas\OpenCV\cvaux\include
$(INC_OPENCV_CVCAM)	C:\Arquivos de programas\OpenCV\otherlibs\cvcam\include

O cubicles tem uma referência para o arquivo "unistd.h" que eu não encontrei no Studio, por isso, usei o arquivo do Dev-C++, adicionando nesse mesmo lugar o diretório:

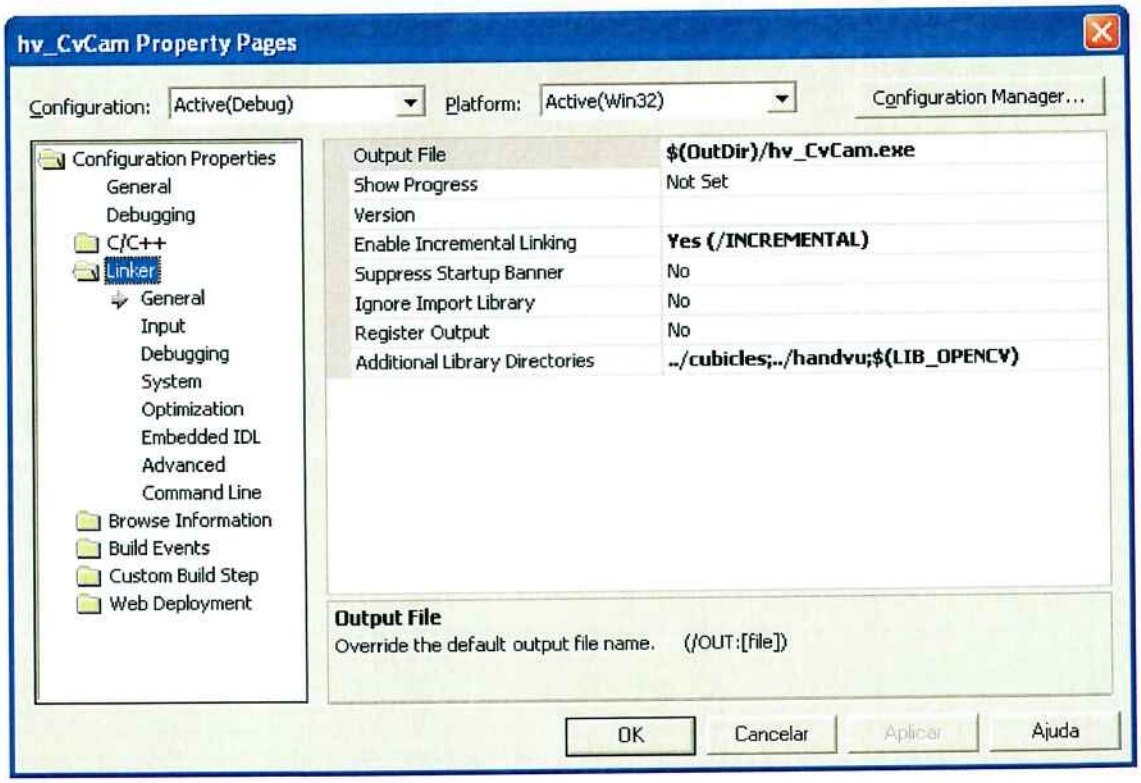
C:\Dev-Cpp\include\sys

Vocês também vão encontrar esse arquivo na pasta "C:\Dev-Cpp\include", mas esse arquivo não serve.

É necessário colocar as referencias para compilação no modo Debug e no modo Release.

Para compilar o exemplo do HandVu (hv_CvCam) também é necessário colocar as referências dos header, seguindo a tabela acima. E também as libs dependentes, na mesma janela de propriedades do projeto, na aba Configuration Properties -> Linker -> General -> Additional Library Directories, colocar a referência do lib do OpenCv.

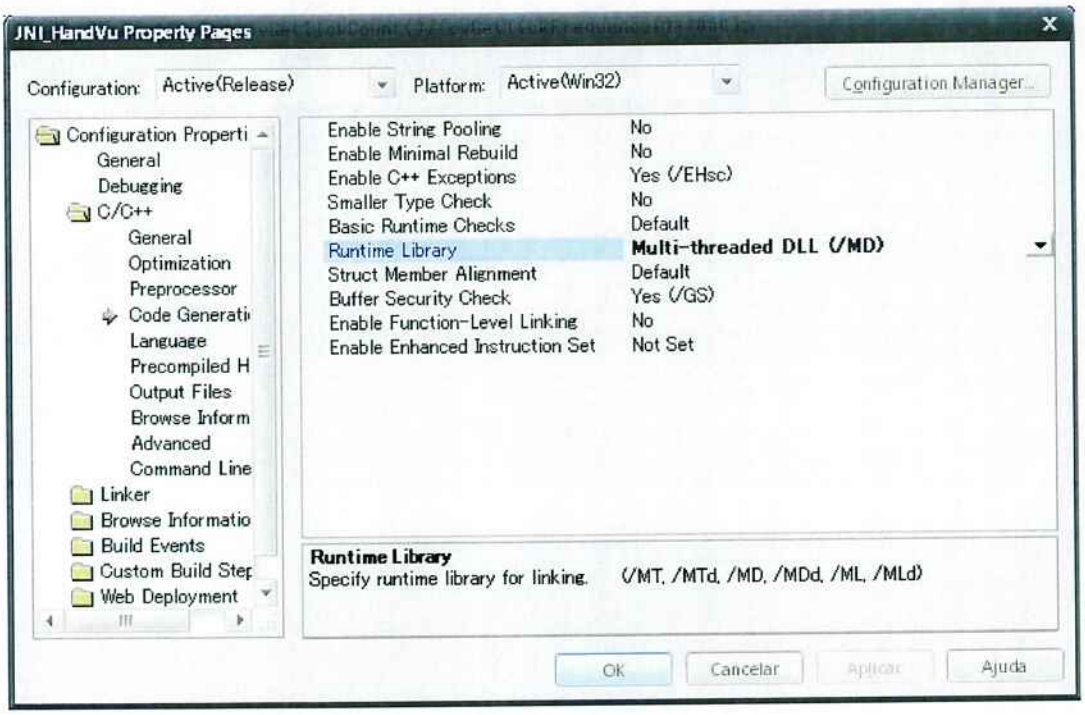
Variavel	Caminho
$(LIB_OPENCV)	C:\Arquivos de programas\OpenCV\lib



Obs.: Depois de adicionar todas as referências, é necessário excluir as variáveis da lista.
Obs.: Tentei fazer isso no Studio 2005 e não funciona da mesma maneira, acho que é devido à conversao do projeto do 2003 para o 2005.

Criação do DLL do HandVu JNI.

Na geração do Dll utilizando o HandVu é necessário alterar mais um parâmetro do sistema. Para isso, em Project -> (nome do projeto) Properties. Na nova janela ir na aba Configuration Properties -> C/C++ -> Code Ganeration. Altere o campo **Runtime Library** para Multi-threaded DLL (/MD) ou para Multi-threaded Debug DLL (/MDd). Como mostra a figura abaixo.



APÊNDICE B – RESULTADO DOS TESTES DE USABILIDADE

A seguir podem ser vistos os resultados obtidos as tarefas do teste informal de usabilidades realizado.

	Usuário			
	1		2	3
	HandVu	HandVu e Teclado	HandVu	HandVu
Tarefa 1				
Número de erros	13	0	2	6
Tempo para realização da tarefa(s)	377	86	108	217
Número de passos	53	38	42	57
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 2				
Número de erros	1	0	1	2
Tempo para realização da tarefa(s)	78	58	36	36
Número de passos	19	27	16	12
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 3				
Número de erros	4	1	4	5
Tempo para realização da tarefa(s)	117	93	96	206
Número de passos	34	30	32	53
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 4				
Número de erros	1	0	0	1
Tempo para realização da tarefa(s)	31	89	17	24
Número de passos	8	29	8	12
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 5				
Número de erros	2	0	2	2
Tempo para realização da tarefa(s)	126	25	20	19
Número de passos	23	6	22	13
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 6				
Número de erros	3	2	0	1
Tempo para realização da tarefa(s)	78	51	44	41
Número de passos	19	16	12	16
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 7				

Número de erros	21	2	4	3
Tempo para realização da tarefa(s)	411	99	181	159
Número de passos	118	29	47	62
Usuário realizou a tarefa?	sim	sim	sim	sim

Tabela 1 – Resultados das tarefas do teste

	Usuário			
	4		**5**	
	Teclado e Mouse	HandVu e Teclado	Teclado	HandVu
Tarefa 1				
Número de erros	2	0	3	24
Tempo para realização da tarefa(s)	42	12	49	300
Número de passos	23	11	22	85
Usuário realizou a tarefa?	sim	sim	sim	desistiu
Tarefa 2				
Número de erros	0	2	0	3
Tempo para realização da tarefa(s)	26	20	8	72
Número de passos	20	12	6	20
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 3				
Número de erros	3	2	0	5
Tempo para realização da tarefa(s)	70	39	4	160
Número de passos	44	25	0	41
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 4				
Número de erros	1	5	0	1
Tempo para realização da tarefa(s)	23	11	16	25
Número de passos	14	5	8	10
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 5				
Número de erros	0	0	0	4
Tempo para realização da tarefa(s)	13	8	12	54
Número de passos	8	6	9	14
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 6				
Número de erros	0	0	0	3
Tempo para realização da tarefa(s)	30	15	25	35
Número de passos	16	10	16	43

Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 7				
Número de erros	1	2	0	7
Tempo para realização da tarefa(s)	41	44	23	137
Número de passos	27	25	15	39
Usuário realizou a tarefa?	sim	sim	sim	sim

Tabela 2 – Resultados das tarefas do teste

	Média			
	Teclado e Mouse	HandVu e Teclado	Teclado	HandVu
Tarefa 1				
Número de erros	2	0	3	11,25
Tempo para realização da tarefa (s)	42	12	49	250,5
Número de passos	23	11	22	59,25
Usuário realizou a tarefa?	sim	sim	sim	sim*
Tarefa 2				
Número de erros	0	2	0	1,75
Tempo para realização da tarefa (s)	26	20	8	55,5
Número de passos	20	12	6	16,75
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 3				
Número de erros	3	2	0	4,5
Tempo para realização da tarefa (s)	70	39	4	144,75
Número de passos	44	25	0	40
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 4				
Número de erros	1	5	0	0,75
Tempo para realização da tarefa (s)	23	11	16	24,25
Número de passos	14	5	8	9,5
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 5				
Número de erros	0	0	0	2,5
Tempo para realização da tarefa (s)	13	8	12	54,75
Número de passos	8	6	9	18
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 6				
Número de erros	0	0	0	1,75

Tempo para realização da tarefa (s)	30	15	25	49,5
Número de passos	16	10	16	22,5
Usuário realizou a tarefa?	sim	sim	sim	sim
Tarefa 7				
Número de erros	1	2	0	8,75
Tempo para realização da tarefa (s)	41	44	23	222
Número de passos	27	25	15	66,5
Usuário realizou a tarefa?	sim	sim	sim	sim
* Um usuário desistiu devido a dificuldade de realização da tarefa por causa de um bug no sistema que já foi corrigido para os próximos testes				

Tabela 3 – Media dos resultados das tarefas do teste

Após a realização das tarefas, os usuários responderam o questionário pós-teste, e os resultados podem ser vistos a seguir:

	Usuários			
	1		**2**	**3**
	HandVu	**HandVu e Teclado**	**HandVu**	**HandVu**
Facilidade de uso	2	2	5	4
Facilidade de navegação	3	4	4	3
Utilidade	3	3	5	4
Facilidade de aprendizado	3	4	4	4
Facilidade de se recuperar de erro	1	4	3	4
Avaliação geral	4	3	5	4

Tabela 4 – Questionário pós-teste

	Usuários			
	4		**5**	
	Teclado e Mouse	**HandVu e Teclado**	**Teclado**	**HandVu**
Facilidade de uso	4	4	4	2
Facilidade de navegação	4	5	4	4
Utilidade	4	4	4	3
Facilidade de aprendizado	4	3	5	2
Facilidade de se recuperar de erro	2	4	4	2
Avaliação geral	4	4	4	2

Tabela 5 – Questionário pós-teste

	Média			
	Teclado e Mouse	**HandVu e Teclado**	**Teclado**	**HandVu**
Facilidade de uso	4	4	4	3,25
Facilidade de navegação	4	5	4	3,5
Utilidade	4	4	4	3,75
Facilidade de aprendizado	4	3	5	3,25
Facilidade de se recuperar de erro	2	4	4	2,5
Avaliação geral	4	4	4	3,75

Tabela 6 – Media dos questionários pós-teste