UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
Departamento de Electrotécnia e Computadores

INTERACÇÃO H/M BASEADA NUMA LUVA

Pedro Miguel Gotelib da Costa Veloso Nogueira

Porto, Julho de 1993

Parecer

Confirmo tanto o empenho revelado no estágio pelo aluno do 5º ano de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto, **Pedro Miguel Gotelib da Costa Veloso Nogueira**, como a qualidade técnica do trabalho realizado "Interacção H/M baseada numa Luva".

Porto e INESC, 28 de Julho de 1993

José Manuel Magalhães Cruz
Assistente da FEUP

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores
Rua dos Bragas, 4099 Porto Codex, PORTUGAL
Telef. 351-2-317105/107/412/457 · Telex 27323 FEUP P · Telefax 351-2-319280

Parecer

Confirmo tanto o empenho revelado no estágio pelo aluno do 5º ano de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto, **Pedro Miguel Gotelib da Costa Veloso Nogueira**, como a qualidade e o interesse científico do trabalho realizado e intitulado "Interacção H/M baseada numa Luva".

Porto e FEUP, 28 de Julho de 1993

Raul Fernando Almeida Moreira Vidal
Professor Associado da FEUP

Interacção Homem/Máquina baseada numa luva

Pedro Miguel Gotelib da Costa Veloso Nogueira
FEUP
1993

AGRADECIMENTOS

Agradeço aos meus orientadores, Prof. Raul Vidal, orientador pela FEUP, e Prof. José Manuel Magalhães Cruz, orientador pelo INESC, por todo o seu apoio prestado ao longo da realização do trabalho.

Gostaria de prestar um agradecimento muito especial ao Prof. Nunes Ferreira, que se mostrou sempre disponível, contribuindo de uma forma eficaz para a elaboração deste trabalho.

ÍNDICE

1. INTRODUÇÃO

Há uma tentativa de transformar o modo como o utilizador interactua com o sistema. Um dos modos de atingir este objectivo está a ser alcançado através da *Realidade Virtual*.

A *realidade virtual* pretende aproximar, o mais correctamente possível, um sistema da realidade que ele simula. Tem como objectivo máximo conseguir que o utilizador não consiga distinguir o sistema, com o qual interactua, do ambiente do utilizador.

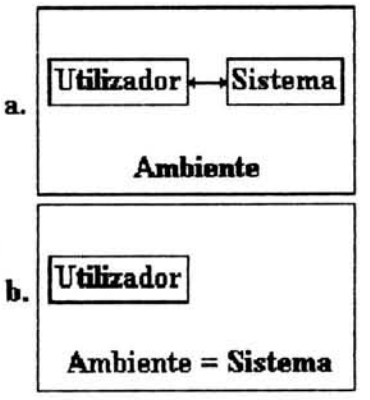

Figura 1. (a)Interface convencional
(b) Interface com Realidade Virtual

A *RV* pode igualmente simular um sistema não compatível com a realidade, mas de forma a atingir os fins propostos. É o caso de certas aplicações na medicina onde se simula um ambiente em que a força da gravidade pode ser controlada.

O maior inconveniente na *RV* é o custo do equipamento associado. Existem plataformas para caminhar, ratos 3D, e os mais habituais, os óculos e as luvas. Dado o custo associado, um sistema de *RV* utiliza apenas alguns dos elementos associados.

Como inconveniente deste tipo de sistemas está o tempo

que demora a desenvolver o software. Tempos desde 6 a 18 meses são vulgares. Tal é devido em grande parte à construção e animação do ambiente. Dadas as exigências de realismo, os objectos têm de ser muitas vezes melhorados até se atingir um resultado aceitável. Além disso constata-se não ser tão fácil utilizar rotinas préviamente feitas como o é nas aplicações convencionais. A produção de rotinas reutilizáveis não parece estar a obter o mesmo êxito que foi demonstrado na programação clássica.

Apesar dos seus inconvenientes a *RV* tem já vários campos de aplicação, onde pode ser eficazmente utilizada: jogos e entretenimento, comunicação à distância, simulação e treino, e visualização.

Mais vulgarmente registamos o seu uso nos jogos e em simuladores.

Numa escola médica em Kentucky, foi desenvolvida uma interface de *RV* para instrumentos avançados. Até à data conseguiram simular em *RV* uma operação cirúrgica completa. Outras aplicações situam-se na área da fisioterapia, na qual os movimentos do corpo humano são profundamente estudados.

A simulação é uma das áreas de preferência da *RV*. O uso de um simulador permite o ensino de uma determinada arte que de outro modo poderia ser muito mais dispendioso, perigoso ou mesmo impossível.

Realizam-se investigações no desenvolvimento de simuladores de condução e de simuladores de aviação. A criação de um sistema virtualmente ligado à realidade permite uma aprendizagem mais correcta em que as acções do utilizador controlam o desenrolar do programa, e não o inverso.

O equipamento utilizado pretende receber todas as entradas provenientes do utilizador necessárias e também interactuar com este enviando-lhe estímulos.

O tapete permite ao utilizador andar sobre ele fornecendo ao sistema o seu movimento sobre um determinado

ambiente. Tem a vantagem de conseguir que o utilizador não se mova do seu local, embora ele sinta que se está a deslocar através da realidade criada pelo computador.

O rato 3D adiciona mais uma dimensão relativamente aos ratos correntes.

Os óculos são constituídos por um ecrã cilindrico que cobre toda a área de visão do utilizador de modo a que ele se abstraia do ambiente real que o rodeia e só tenha a percepção do ambiente criado.

As luvas são bastantes úteis. Têm a grande vantagem de serem de compreensão imediata. Permitem funcionar como um rato 3D, mas são muito mais abrangentes através do uso dos dedos para a realização de operações mais reais (manipulação de objectos). Algumas permitem ainda receber uma resposta do computador de modo a simularem as sensações de tacto e de força. Tal é feito limitando o movimento dos dedos.

É uma área em grande expansão, da qual se esperam nos próximos anos grandes desenvolvimentos.

2. OBJECTIVOS

Pretende-se com o trabalho realizado fazer uma introdução ao ambiente da realidade virtual.

Torna-se necessário a criação de uma série de plataformas base que sirvam de apoio ao desenvolvimento de acções de manipulação de objectos.

De modo a atingir este objectivo foram enumerados os seguintes sub-objectivos:

- Estruturas de dados para visualizar uma cena 3D
  - Visualização no ecrã do esqueleto da mão
  - Visualização no ecrã de objectos simples

- Remoção da cena de faces escondidas

- Iluminação da cena

- Interacção com a Power Glove
  Movimentar a luva no ecrã movendo a mão e os dedos (através da Power Glove)

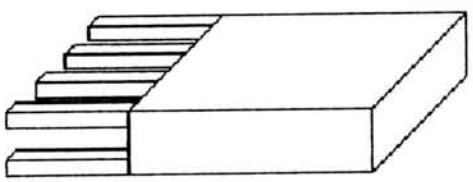

Figura 2. Esqueleto da mão

3. ESTRUTURAS DE DADOS

3.1 Descrição das estruturas

As estruturas de dados a adoptar para a descrição do mundo foram por diversas vezes alteradas.

Inicialmente foi adoptada uma estrutura do tipo *wireframe*, na qual as faces são transparentes e apenas são visualizadas as arestas.

Para uma estrutura deste tipo podemos adoptar a seguinte representação:

Cena
  Sólido
    Aresta
      Vértices

No entanto, este tipo de representação mostra não ser a mais adequada para uma boa visualização, dado que não permitia saber com precisão a orientação de um objecto no espaço. Duas orientações diferentes eram mapeadas na mesma imagem no ecrã.

Além disso esta estrutura era extremamente ineficiente na forma de extrair informações sobre as faces, o que seria necessário na segunda fase do trabalho.

Adoptou-se então uma nova representação mais adequada, que é representada a seguir:

Cena
  Sólido
    Faces
      Vértices

Esta representação permite a manipulação das faces de uma forma imediata. No entanto, foi retirada a

representação das arestas pois exigiria mais um nível na representação hierárquica.

Verifica-se que um grande número de níveis é inconveniente pois dificulta a manipulação dos dados, torna o código pouco claro e longo, e aumenta o tempo de processamento. Deste modo, uma representação consistindo de 3-4 níveis é preferível a uma representação demasiado pormenorizada.

A representação que foi finalmente escolhida consiste numa pequena transformação da anterior, de modo a retirar toda a informação redundante. Se analizarmos a representação anterior veremos que um vértice é repetido três vezes, já que pertence a três faces simultâneamente. Assim, se as contas forem realizadas a nível das faces far-se-ão três vezes mais cálculos do que o necessário.

A maneira de obviar a este gasto de tempo de processamento consiste em representar numa estrutura Vértices todos os vértices do sólido e na estrutura Faces os apontadores para esses vértices.

Os cálculos são todos feitos na estrutura Vértice, de modo que cada ponto é apenas calculado uma vez.

A representação final terá a seguinte representação:

Cena
  Sólido
    Faces
    Vértices

3.2 Representação das estruturas

vector

float. Array de dimensão 3
Representa {x,y,z}

TransfMat

Matriz 3x3

point_3D

Array de dimensão 4
Representa {x,y,z,w}

point_img

Array de dimensão 3
Representa {x,y,z}

observers

Campos	Domínio e descrição
rotx	float. Rotação em x do observador
roty	float. Rotação em y do observador
rotz	float. Rotação em z do observador
ro	float. Deslocação em z do observador
focus	float. Distância do foco
Trans	TransfMat. Matriz de transformação do observador. Incorpora as rotações e o deslocamento.

face

Campos	Domínio e descrição
num_vertices	short. Número de vértices da face
vertex	short. Array com os apontadores para o array vertex da estrutura solid
normal_world	vector. Vector normal à face
normal_obs	vector. Vector normal à face

face_img

Campos	Domínio e descrição
num_verts	short. Número de vértices
vert_img	short. Array com os apontadores para o array vertex da estrutura solid

solid

Campos	Domínio e descrição
transl	vector. Translacção do objecto
rotx	float. Rotação em x do objecto
roty	float. Rotação em y do objecto
rotz	float. Rotação em z do objecto
num_verts	short. Número de vértices do objecto
num_faces	short. Número de faces do objecto
num_vis_faces	short. Número de faces visíveis
vertex	point_3D. Array que contém os vértices na sua posição inicial no mundo
vert_world	point_3D. Array que contém os vértices na sua posição actual no mundo
vert_obs	point_3D. Array que contém os vértices na sua posição vista pelo observador
vert_img	point_img. Array com o valor dos vértices a serem colocados no ecrã
faces	face. Definição de cada face
vis_faces	short. Array que contém apontadores para as faces vísiveis no array faces
color	short. Cor dominante do objecto

vis_fac_order

Campos	Domínio e descrição
xmax	float. Valor máximo em x dessa face
xmin	float. Valor minimo em x dessa face
ymax	float. Valor máximo em y dessa face
ymin	float. Valor minimo em y dessa face
zmax	float. Valor máximo em z dessa face
zmin	float. Valor minimo em z dessa face
ind_solido	short. Índice do objecto em causa
ind_face	short. Índice da face em causa

4. DESCRIÇÃO DE ALGORITMOS

4.1 Mudanças de base

Na cena a representar existem vários sistemas de eixos, sendo possível através de transformações passar de uns para os outros. Estas transformações, dado estarmos em presença de um espaço tridimensional consistem em valores de rotação em torno de cada um dos eixos, x, y, z, e num vector de translacção.

São definidos vários sistemas de eixos existentes na cena.

O *sistema de eixos absoluto*, aquele que serve de base para todos os outros, designamos pelo sistema de eixos do mundo. É relativamente a este sistema que os objectos são modelizados, sendo posteriormente transformados de modo a localizarem-se nas suas posições e orientações actuais.

É também relativamente a este sistema que os restantes são definidos.

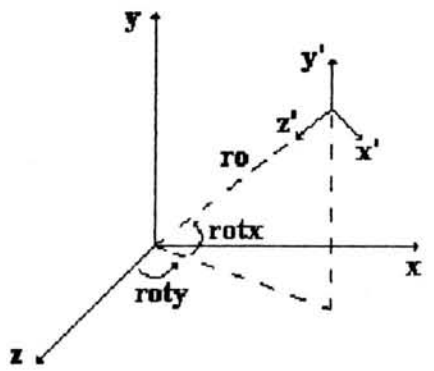

Figura 3. Sistema de eixos do mundo/obsevador

Outro sistema de eixos muito importante é o sistema de eixos do observador. É este sistema de eixos que define a imagem observada no ecrã, ou seja o que é visto pelo observador.

Este sistema não apresenta os habituais seis graus

de liberdade, mas apenas quatro. Permitem-se definir relativamente ao sistema de eixos do mundo o valor das rotações em torno de cada um dos eixos e o valor da deslocação em relação ao eixo dos zz.

Deste modo e como se pode constatar pela figura 3, o eixo dos zz do observador fica sempre orientado para a origem do sistema de eixos do mundo. Pretende-se com esta restrição que o observador esteja permanentemente a visualizar a cena.

Os restantes sistemas de eixos existentes estão ligados aos objectos.

Tem particular interesse o sistema ligado à luva, o qual está centrado na *palma* da luva. Este sistema é utilizado pelos *dedos* da luva, cujas coordenadas se referem a ele. Deste modo os *dedos* ficam ligados à *palma* da luva sendo influenciados por qualquer movimento efectuado por esta.

As matrizes de transformação são apresentadas a seguir.

$$\begin{vmatrix} \cos\theta & -sen\theta & 0 & 0 \\ sen\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -sen\varphi & 0 \\ 0 & sen\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} \cos\phi & 0 & sen\phi & 0 \\ 0 & 1 & 0 & 0 \\ -sen\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

(a) **(b)** **(c)**

$$\begin{vmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

(d)

Figura 4. Matrizes de (a) rotação em z; (b) rotação em x; (c) rotação em y e (d) translacção

A passagem de uma determinada base para outra é feita à custa de uma *matriz de transformação*. Na

realidade temos mais do que uma matriz de transformação: três matrizes relacionadas com a rotação em torno de cada um dos eixos x, y e z e uma matriz de translacção em relação à origem.

De modo a efectuar a mudança de base efectua-se a seguinte multiplicação

Matriz de transformação x ponto = novo ponto

Esta *Matriz de transformação* é uma matriz única e é conseguida multiplicando as matrizes anteriores pela ordem seguinte

$$Matriz_{transl} \times Matriz_{rotz} \times Matriz_{roty} \times Matriz_{rotx}$$

Esta concatenação de matrizes permite uma maior velocidade de processamento, já que apenas será efectuada uma multiplicação de matrizes, em vez de quatro, para cada ponto.

Dado que em *realidade virtual* o tempo é essencialmente gasto em manipulação de dados, esta vantagem é fundamental.

4.1.1 *Matriz de transformação do observador*

Dado que o observador tem apenas quatro graus de liberdade, o valor dos deslocamentos em x e y são nulos. Virá, portanto, na matriz de translacção,

$$p_x = p_y = 0$$

A *matriz de transformação* do observador tem ainda uma particularidade. Dado que pretendemos que o observador fique orientado para a origem da *base do mundo* é necessário proceder a uma rotação em z de 180º. Senão vejamos, se o valor das rotações em torno de cada

um dos eixos forem nulas, ou seja

$$\theta = \varphi = \phi = 0$$

o sistema de eixos do observador permanecerá idêntico ao sistema de eixos do mundo. A consequente translacção em z manterá o sistema de eixos do observador orientado no sentido inverso à origem, como pretendíamos. Deste modo, antes de se efectuar a translacção torna-se necessário realizar uma rotação extra de 180º em torno de z.

Esta *matriz* é aplicada a cada um dos objectos definidos nas suas posições actuais, ou seja depois de cada um deles ter sofrido a transformação da sua própria *matriz de transformação*. Deste modo obteremos a imagem dos pontos visualizados pelo observador.

Bastará agora proceder à aplicação da perspectiva para que os pontos possam ser impressos no ecrã.

4.1.2 *Matriz de transformação dos objectos*

Cada objecto criado na cena tem a sua própria *matriz de transformação*, a qual conterá os valores dos parâmetros de rotação e de translacção que definem a posição e orientação desse objecto no espaço.

É importante lembrar que uma *matriz de transformação* tem como objectivo a mudança de base, logo devendo ser aplicada relativamente à base a que diz respeito. Deste modo há que ter em conta que os *dedos* têm como referencial o sistema de eixos da *palma*.

O cálculo da sua localização é feita relativamente a este referencial. Portanto, realizam-se desta vez duas multiplicações matriciais, uma para passar do referencial do mundo para o referencial da *palma* e outra para passar deste referencial para o dos *dedos*.

4.2 Remoção de faces escondidas

Até este momento podemos representar no ecrã todos os objectos criados, podendo haver ligações entre eles. No entanto teremos de nos limitar a uma visualização em *wireframe*, visto que todas as faces são desenhadas.

Como já foi referido, esta representação em *wireframe* tem a desvantagem de enganar o observador quanto à verdadeira orientação do objecto. Além disso, com a presença de vários objectos na cena é dificil perceber a localização relativa de uns em relação aos outros. Essa dificuldade é aparente quando dois objectos se encontram muito próximos e é quase impossível saber qual deles está à frente de qual.

Apesar de que o abandono duma estrutura do tipo *wireframe* significa um sacrifício em tempo de processamento, constata-se que os ganhos obtidos são por demais satisfatórios.

Pretendemos conhecer, dado um conjunto de objectos 3D e uma direcção de visualização, quais as superfícies dos objectos que estão visiveis, total ou parcialmente, de modo a que possamos desenhar apenas essas.

Dado o custo de processamento que envolve a determinação destas superfícies, percebe-se agora melhor a necessidade apontada inicialmente em adoptar a estrutura de dados que melhor se adaptasse a este algoritmo.

Se a dificuldade for o tempo de processamento então devemos sacrificar em memória, mesmo que tal signifique o ganho de redundância.

Existem dois tipos de algoritmos: algoritmos orientados à imagem e algoritmos orientados ao objecto.

Os algoritmos orientados à imagem são realizados ao nível de resolução do monitor utilizado. Traduzem-se em varrer toda a imagem e cálcular para cada pixel da imagem a sua visibilidade. Estes algoritmos dão óptimos

resultados, mas são muito lentos.

Os algoritmos orientados ao objecto são realizados com a precisão com que cada objecto foi definido e determinam a visibilidade de cada objecto. Relativamente ao outro tipo de algoritmos precisam de uma etapa final que ajuste os objectos à resolução do monitor.

Como a cena a representar é constituida por vários objectos não é suiciente conhecer de cada objecto quais as faces visíveis, mas também a ordem pela qual iremos desenhar todas as faces. Tal deve-se ao facto de umas faces irem sobrepor-se a outras pela razão de se situarem mais próximas do observador.

O método que foi utilizado consistiu em aplicar o algoritmo 'Back-Face Culling' a cada objecto de modo a determinar para cada um as faces visíveis. Em seguida aplica-se o *algoritmo do pintor* de modo a ordenar as faces anteriormente seleccionadas.

4.2.1 *Back-Face Culling*

Uma face não visível tem a propriedade de um vector normal à sua superfície, calculado pela *regra da mão direita*, ter uma componente em z negativa.

Isto significa que está orientado no sentido oposto ao plano de visualização e a face corrente será totalmente coberta por outras faces desse políedro. Removendo as faces que apresentarem esta característica ficaremos apenas com aquelas que são visíveis, como era pretendido.

4.2.2 *Algoritmo do Pintor*

Este algoritmo insere-se num conjunto de algoritmos designados por *List-priority algorithms*. São algoritmos que determinam uma ordenação de visibilidades,

comprometendo-se a fornecer uma imagem correcta se a imagem for construída por aquela ordem.

Por exemplo, se nenhum objecto se sobrepõe a outro em z, então só precisamos de ordená-los por ordem crescente de z, e desenhá-los nessa ordem.

Se um objecto ficar sobreposto a outro em z, é ainda possível calcular a sua ordem à custa de uma série de condições. Se os dois ou mais objectos se sobreposerem ciclicamente um ao outro, ou se se penetrarem, não há uma ordem correcta. Neste caso, ter-se-ia que partir um dos objectos pela intersecção do outro e tratar cada uma das suas partes como objectos independentes.

Esta situação não é tratada pelo algoritmo já que os objectos usados na cena são convexos e como tal esta situação nunca ocorrerá.

As etapas pelas quais passa o algoritmo são as seguintes:

1. Ordenar todos os polígonos pela maior coordenada em z de cada um
2. Resolver as ambiguidades causadas pela sobreposição em z, trocando polígonos se necessário
3. Desenhar cada polígono por ordem descendente da maior coordenada em z

De modo a resolver a segunda etapa, necessitámos de testar os dois polígonos em questão. Tal é feito recorrendo-se a uma série de testes de complexidade ascendente. Seja P o polígono situado no fim da lista ordenada. Para que este seja correctamente desenhado é necessário provar que não tapa qualquer outro polígono Q dessa lista. Seja Q um polígono que se sobrepõe a P em z. Os polígonos P e Q terão de ser testados. Assim que um dos testes suceder, ou seja for verdadeiro, concluímos que P não tapa Q e passaremos ao próximo

polígono Q que se sobreposer a P em z.

Os cinco testes são brevemente expostos a seguir:

1) Os polígonos não se sobrepõem em x?
2) Os polígonos não se sobrepõem em y?
3) Está P totalmente em frente de Q?
4) Está Q totalmente em frente de P?
5) As projecções dos dois polígonos no plano (x,y) não se sobrepõem?

O teste 3 pode ser feito calculando a equação do plano de Q e substituindo nela todos os vértices de P. Se todos eles forem positivos então o teste sucede.

O teste 4 será executado de forma semelhante. Se este teste suceder a ordem dos dois polígonos na lista deve ser trocada.

Se todos os testes falharem concluí-se que P tapa Q e devemos voltar a testar 3, 4 e 5, mas trocando Q com P. Se estes testes voltarem a falhar estamos numa situação em que não há uma ordem correcta para os desenhar. Um deles terá de ser cortado.

4.3 Iluminação

De modo a tornar mais perceptível as superfícies planas dos objectos torna-se necessário pintar cada face de um objecto com uma cor diferente.

Deste modo podemos reconhecer fácilmente a geometria de um determinado objecto, aumentando o realismo e a clareza da cena visualizada.

Pretendia-se com a iluminação atingir objectivos muito específicos, de modo que adoptou-se uma configuração muito simplificada.

A simplificação não retira qualquer validade ao algoritmo, mas ganha-se na relação entre o tempo de processamento necessário e os resultados obtidos.

A resolução escolhida para o monitor foi VGA. Escolheu-se o modo de duas páginas, que permite um maior realismo de animação, mas apenas permite manipular 16 cores.

Como tal, o número de tons diferentes de cada cor não excede quatro, sendo totalmente suficientes para o tipo de objectos com que lidamos. Um número de três tonalidades diferentes teria sido igualmente aceitável.

Manipulando a palete de cores, obtivemos a seguinte tabela:

0	Preto
1	Branco
2,3	dois tons de cinzento
4 a 7	quatro tons de azul
8 a 11	quatro tons de verde
12 a 15	quatro tons de vermelho

Tabela 1. Tabela de cores

A fonte de iluminação não consiste num só ponto, mas num plano de luz paralelo ao plano xz. Tal

significa que qualquer vector de luz está completamente determinado, tendo componente nula em x e em z, e componente negativa em y.

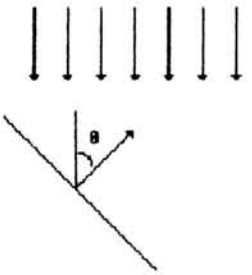

Figura 5. Iluminação de uma superfície por um plano de luz

A cada objecto está associada uma cor, de modo que as suas faces irão ser pintadas por uma das tonalidades disponíveis. A escolha da tonalidade a atribuir a uma face é feita calculando o ângulo formado entre um vector de luz e a normal da superfície dessa face, como pode ser observado na figura 5.

A atribuição é feita segundo o seguinte intervalo de valores,

Ângulo formado	Tonalidade
0 a 30	cor1
30 a 45	cor2
45 a 60	cor3
60 a 90	cor4

Tabela 2. Atribuição de tonalidades

5. HARDWARE

O programa foi desenvolvido para PC.

Adoptou-se a resolução de VGA, no modo de duas páginas. Este modo permite uma melhor animação, dado que permite a criação da cena de um modo invisível ao utilizador.

A página activa é sempre a não visível, de modo que o utilizador não se apercebe do desenho de toda a cena. Quando a cena estiver totalmente construída, esta página passa a ser visível, mostrando ao utilizador o resultado de nova iteracção. A página activa passa então a ser a outra página, não visível, repetindo-se o processo.

A ligação à luva foi feita à custa de outro software desenvolvido, o qual tem por missão 'compreender' e 'traduzir' os valores recebidos pelo hardware.

Os valores transmitidos ao programa eram os seguintes:

- posição da luva
- orientação da luva em torno do eixo dos zz
- inclinação dos dedos

Os valores transmitidos ao programa permitem que este programa seja independente de qualquer luva ou outro dispositivo (apenas do PC). A ligação a outro tipo de luva é possível alterando-se apenas o software de interface.

O programa permite ainda aceitar todos os comandos através do teclado, dando prioridade aos valores recebidos através da luva.

6. CONCLUSÕES

Os objectivos propostos foram atingidos.

Obteve-se uma plataforma para a construção de cenas tridimensionais, constituidas por sólidos convexos. Esta convexidade é exigida pelo algoritmo de eliminação de faces escondidas.

A estrutura de dados aceita qualquer tipo de objectos de faces planas. A descrição de objectos não convexos pode provocar na cena incoerências com a realidade, já que uma face que atravesse outra pode, em determinadas circunstâncias, aparecer à frente desta.

O algoritmo trata já vários tipos de sobreposição, mas não executa o corte de faces, o que pode ser essencial em objectos não convexos.

A ligação à luva foi conseguida com grande facilidade. Infelizmente apenas um dos valores de rotação eram fornecidos, logo a luva não permitia explorar todas as capacidades do programa.

Futuramente o programa pode ser expandido de modo a permitir a manipulação de objectos existentes na cena. Deste modo poderemos concluir que o programa entra na área da *realidade virtual* através da realização de operações sobre objectos virtualmente criados.

7. BIBLIOGRAFIA

"Fundamentals of Interactive Computer Graphics"
J.D. Foley, A. Van Dam
Addison-Wesley, 1990

"Computer Graphics"
Donald Hearn, M. Pauline Baker
Prentice Hall

"Computational Geometry, an introduction"
Franco P. Preparata and Michael Ian Shamos
Springer-Verlag 1985

"Advances in Computer Graphics II"
R.J. Hubbold, and D.A. Duce
Edited by F.R.A. Hopgood

"Computer Gems"
Academic Press

Spooler Magazine N°1