Interacção com um Cubo de Rubik Virtual

Carlos Silva, Milton Ruas

Abstract – **There are several ways to manipulate a virtual Rubik's Cube, either using a large number of keys from a PC's keyboard (inefficient for large cubes) or bringing the mouse into action. In this paper we present an efficient method to manipulate the cube using the mouse (or mouse and keyboard).**

Resumo – **Há muitas formas de manipular um Cubo de Rubik virtual, desde métodos que impliquem o uso de grande parte das teclas disponíveis no teclado (bastante ineficiente para cubos grandes) a métodos que envolvam também o uso do rato. Neste artigo apresentamos uma forma eficiente de manipulação do cubo com o rato (isolado ou em combinação com o teclado).**

I. HISTÓRIA

O *Cubo de Rubik* nasceu na Hungria em 1974 através de Erno Rubik, cuja paixão pela arte e pela técnica o levou à criação de um jogo em que o oponente é a própria natureza. A curiosidade pelo espaço e a relação deste com o Homem, que está na base da sua formação em Arquitectura e Design, levou-o a pensar na figura geométrica do cubo [1]. Desde a sua origem até hoje, este jogo tem atraído inúmeras pessoas, existindo alguns campeonatos por todo o mundo. Actualmente existem várias versões do Cubo de Rubik destinadas aos apaixonados deste objecto, das quais se salientam o cubo de $5 \times 5 \times 5$ e o cubo 4-D.

Existem vários métodos para resolução do Cubo de Rubik, todos eles resultando da Teoria de Grupos [2]. Esta foca o estudo da simetria de forma abstracta e fornece uma ligação entre o espaço e a estrutura. As aplicações desta área da Matemática são vastas: Cristalografia, Mecânica Quântica, Física Molecular, etc.

O Cubo de Rubik é, desta forma, uma aplicação que permite desenvolver as capacidades intelectuais e de discernimento espacial a quem se propõe resolvê-lo.

II. INTRODUÇÃO

O Cubo de Rubik é um cubo definido por uma cor diferente em cada uma das faces (quando resolvido), sendo constituído por vários cubos mais pequenos. Os cubos pequenos não são fixos, podendo deslocar-se com alguma independência em relação aos outros, mas nunca individualmente. O deslocamento é feito pela rotação de uma linha ou coluna de cubos (discos cúbicos) em relação a qualquer um dos eixos tridimensionais. O objectivo do jogo é, pela manipulação dos discos, obter uma só cor em cada face do Cubo de Rubik.

No âmbito da disciplina de Computação Gráfica (LEET, opção de 5º ano), pretendia-se construir uma aplicação que simulasse o tradicional Cubo de Rubik para vários tamanhos diferentes (de $2 \times 2 \times 2$ a $10 \times 10 \times 10$), usando as bibliotecas OpenGL e GLUT [3].

Como o cubo tradicional é essencialmente jogado com as mãos, foi necessário fornecer um método suficientemente inteligente que permitisse que a interacção com o computador fosse o mais intuitiva possível, tornando assim, como veremos, o uso do rato imprescindível.

III. MANIPULAÇÃO GLOBAL DO CUBO

A manipulação do modelo, para permitir a visualização de todas as faces do cubo, é feita de modo a permitir rodar o cubo em torno de qualquer eixo tridimensional.

Estão disponíveis três formas de o fazer:

- Rotações incrementais utilizando o teclado;
- Rotações automáticas de $\pm 90°$, segundo qualquer eixo, utilizando o teclado;
- Rotações automáticas de $\pm 90°$ utilizando o rato.

As rotações automáticas usando o rato são executadas premindo o botão esquerdo com o cursor no exterior do modelo, fazendo com que, dependendo do local, o modelo rode no sentido apropriado. Foram estabelecidas quatro zonas da janela para efectuar estes movimentos (figura 1).

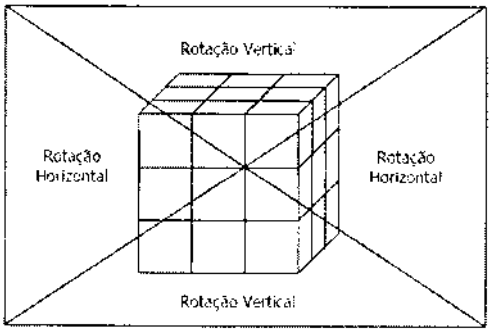

Figura 1 - Zonas para a rotação do modelo usando o rato

A identificação da posição do cursor é feita por verificação das coordenadas do pixel seleccionado relativamente às duas rectas fronteira (verificando se o pixel está acima ou abaixo de cada uma das rectas é possível determinar qual a zona seleccionada).

IV. MANIPULAÇÃO DOS DISCOS CÚBICOS

O Cubo de Rubik, como já foi referido, é constituído por um conjunto de cubos mais pequenos de características próprias. Por isso, na representação interna do modelo do Cubo de Rubik, cada um desses cubos possui um descritor do seu estado, na forma de uma estrutura com os seguintes campos:

Parâmetros de translação local: que indicam a posição relativa do cubo no modelo (figura 3);

Parâmetros de rotação local: que permitem a rotação dos cubos, em torno do centro do modelo, de forma independente dos outros;

Propriedades: de cada uma das faces (componentes de cor).

As propriedades das faces são armazenadas num *array* de seis elementos (seis faces por cubo), cujas referências seguem o modelo da figura 2.

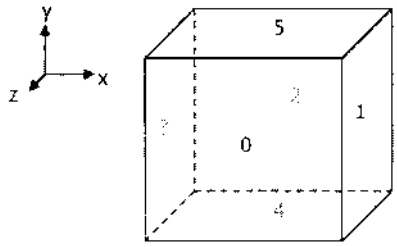

Figura 2 - Referências das faces de cada cubo

Os diversos cubos são organizados num *array* tridimensional de ponteiros que apontam para os descritores dos cubos. Tal escolha deveu-se à simplicidade na referência a um determinado cubo, usando apenas as coordenadas x, y, z (figura 3).

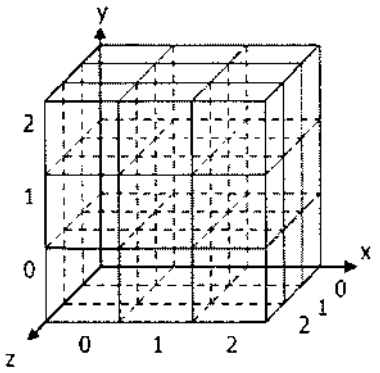

Figura 3 - Coordenadas dos cubos num modelo $3 \times 3 \times 3$

Para a rotação de um disco cúbico, é necessário rodar vários cubos ao mesmo tempo. Por exemplo, no modelo de $3 \times 3 \times 3$ é necessário rodar nove cubos simultaneamente segundo um ângulo de $\pm 90°$.

A primeira abordagem que adoptámos foi, simplesmente, aplicar uma transformação de rotação de $90°$ aos cubos pertencentes ao disco. Apesar de funcionar para o primeiro movimento, apresenta falhas nos seguintes: o deslocamento é feito por referência directa aos cubos a mover do *array* tridimensional, e considerando que a referência de um cubo corresponde à sua posição absoluta segundo a figura 3, logo após o primeiro movimento as referências dos ponteiros já não estão relacionados com as posições dos cubos, conduzindo este método ao fracasso.

De modo a corrigir este problema, decidiu-se evitar o uso de transformações geométricas, e elaborar um algoritmo que contemplasse a actualização dos ponteiros dos cubos. Este contempla quatro passos a executar para cada cubo a deslocar:

1. Cálculo da referência do cubo que irá substituir o actual. Para tal, são efectuadas as atribuições:
   - Para rotações de $+90°$
     $var1(novo) = rubik.ladocubos - 1 - var2(antigo)$
     $var2(novo) = var1(antigo)$
   - Para rotações de $-90°$
     $var1(novo) = var2(antigo)$
     $var2(novo) = rubik.ladocubos - 1 - var1(antigo)$

   $rubik.ladocubos$ indica o número de cubos que o modelo tem de lado (3 para o modelo $3 \times 3 \times 3$); $var1$ e $var2$, são as componentes x, y ou z das coordenadas dos cubos, e dependem do eixo de rotação (figura 3):

	Rotação XX	Rotação YY	Rotação ZZ
$var1$	Y	Z	X
$var2$	Z	X	Y

2. O ponteiro do cubo actual passará a apontar para o novo cubo, que ocupará a posição do cubo actual após o deslocamento;
3. Actualização da posição relativa do novo cubo (alteração dos parâmetros de translação local);
4. Rotação das propriedades das faces. É importante verificar que as faces terão de rodar segundo o mesmo eixo de rotação do cubo, para que as cores correspondam às posições certas depois do deslocamento (figura 2).

Um pormenor importante e que se deve ter em conta, é que a actualização dos ponteiros segue um caminho circular (ao longo do disco a mover), ou seja, ponteiros actualizados serão novamente utilizados para consulta na actualização de futuros cubos (as suas posições antigas passarão a ser as posições de outros cubos), o que representa um problema pois os ponteiros consultados deverão sempre apontar para as posições antigas.

Para resolver esta questão utilizou-se um *array* temporário de dimensões iguais ao primeiro, que inicialmente aponta para os mesmos descritores que o *array* principal. Este *array* será utilizado para consultar as posições antigas dos ponteiros, actualizando apenas o *array* principal.

Com este algoritmo, todos os deslocamentos são feitos com sucesso para cubos de qualquer dimensão.

V. INTERACÇÃO COM O MODELO

Três métodos de interacção com o modelo são possíveis, no que diz respeito à rotação dos discos:

- Usando apenas o teclado;
- Usando apenas o rato;
- Usando simultaneamente o teclado e rato.

A. Interacção usando apenas o teclado

Este é o método mais simples. Foram atribuídas duas teclas, correspondentes à rotação no sentido directo ou inverso, para cada disco de cubos. Usando o modelo de $2 \times 2 \times 2$, temos 6 discos deslocáveis, pelo que são precisas

12 teclas no total, para efectuar todo o tipo de movimentos (figura 4).

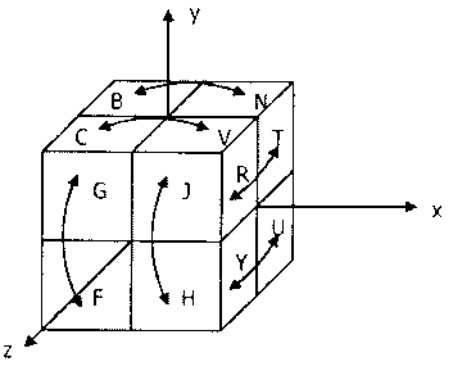

Figura 4 - Manipulação do modelo 2 × 2 × 2 usando apenas o teclado

Este princípio também funciona no modelo de 3 × 3 × 3, contudo apenas manipula os discos extremos. O disco central apenas pode ser deslocado por manipulação simultânea dos discos extremos.

Já a partir do modelo de 4 × 4 × 4, este método torna-se incompleto, pois existem discos que nunca poderão ser rodados, pelo que não foi utilizado a partir destas dimensões.

Apesar de ser um método simples, é pouco intuitivo, dada a elevada quantidade de teclas a usar. Leva também a algum tempo de adaptação por parte do jogador e, além disso, variando a orientação do modelo perde-se a noção das teclas.

B. Interacção usando apenas o rato

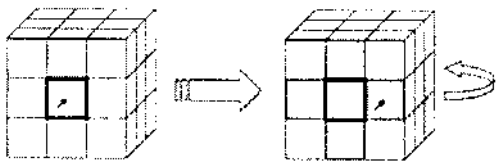

Figura 5 - Rotação dos discos usando apenas o rato

Procedimento:

1. Seleccionar uma face referência – um contorno colorido surge à volta desta face;
2. Seleccionar uma face ao lado, no sentido do deslocamento pretendido.

Este é o método mais simpático ao utilizador, dado que é muito intuitivo e não depende da orientação do modelo. No entanto, a implementação torna-se mais complexa, dado que é necessário relacionar o pixel seleccionado com o objecto desenhado.

Tal complexidade provém do facto de ser necessário percorrer o *pipeline* de visualização ao contrário, tarefa nada fácil de realizar, embora teoricamente possível. O que o OpenGL faz é, em vez de fazer o *rendering* das faces a seleccionar no *color buffer*, fá-lo no *select buffer*, definindo para cada uma delas um identificador. Deste modo, é só seguir as suas posições sempre que são desenhadas, e compará-las com a posição de selecção do rato [4] [5].

Sempre que a posição de uma ou mais faces coincide com a selecção do rato, várias mensagens (uma por cada face seleccionada) são geradas e armazenadas num *stack buffer*. Deste modo, cada mensagem deve ser analisada separadamente e apenas escolhida a face que se encontra mais próxima do utilizador.

A atribuição de um identificador a cada face é feita na forma de um número inteiro, e segue a seguinte expressão (assim cada face possui um identificador único diferente de todos as outros):

$$name = (x \times rubik.ladocubos^2 + y \times rubik.ladocubos + z) \times 6$$

Quando o rato selecciona uma face, a indicação do cubo e da face seleccionada em termos da posição (x, y, z, face) é obtida da seguinte forma:

$$x = \left\lfloor \frac{name}{rubik.ladocubos^2 \times 6} \right\rfloor$$

$$y = \left\lfloor \frac{name}{rubik.ladocubos \times 6} \right\rfloor \% \; rubik.ladocubos$$

$$z = \left\lfloor \frac{name}{6} \right\rfloor \% \; rubik.ladocubos$$

$$face = name \% 6 \qquad \text{(6 faces para cada cubo)}$$

Este método de selecção, apesar de ser bastante fácil de usar, apresenta algumas desvantagens. A principal é que, devido à elevada duração na resolução do Cubo de Rubik, o uso do rato torna-se cansativo.

Uma forma de tentar resolver este problema é o terceiro método de interacção: usando teclado e rato simultaneamente.

C. Interacção usando o teclado e o rato

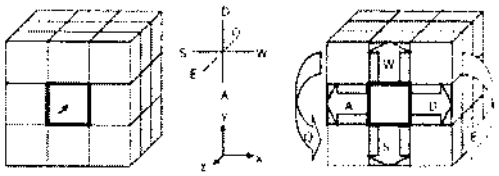

Figura 6 - Rotação dos discos usando teclado e rato

Procedimento:

1. Com o rato, selecciona-se a face referência;
2. Usando o teclado selecciona-se a direcção de deslocamento, pressionando a tecla correspondente.

Este método combina os dois anteriores, e procura minimizar as desvantagens de ambos:

- O número de teclas a usar reduz-se, permitindo uma melhor adaptação do utilizador às teclas;
- A utilização do rato reduz-se para metade pois, em vez de efectuar duas selecções basta fazer uma só.

De modo a minimizar o problema da associação entre as teclas e os discos a rodar, foi incluído um sistema de eixos (figura 6) que tem como propósito ajudar o utilizador: estes eixos representam os eixos de rotação do cubo e, como está na figura, o eixo vertical representa a rotação em torno de YY, com as teclas A e D correspondentes aos sentidos

horário e anti-horário respectivamente. O mesmo se aplica para os outros eixos.

Este foi o método adoptado, pois é o único que indica sempre a informação correcta, qualquer que seja a orientação do modelo, oferecendo melhor jogabilidade quando comparado com os métodos anteriores. Contudo, as vantagens deste método só poderão ser percebidas após alguma adaptação.

VI. *RENDERING* DO MODELO

O *rendering* ("desenho") do modelo, é feito a partir da informação de uma só face. Assim, para a construção de todo o modelo, apenas é necessário efectuar transformações geométricas a cada face, de modo a colocá-las na localização certa. Deste modo, apenas são precisos 4 vértices para desenhar a face, permitindo construir qualquer modelo independentemente da sua dimensão (em cubos).

De modo a maximizar o desempenho computacional, apenas são desenhadas as faces que estão voltadas para o exterior do modelo. Por exemplo, para um modelo de $3 \times 3 \times 3$ são desenhadas apenas 54 faces, em contraposição às 162 faces que o modelo possui na sua totalidade. Esta técnica torna-se indispensável para os imponentes modelos de $10 \times 10 \times 10$.

A. *Desenho do modelo*

As coordenadas dos vértices da face base devem corresponder à face de referência 0 (figura 2) e devem estar colocadas na parte frontal de um cubo de aresta 2 e centrado na origem (figura 7).

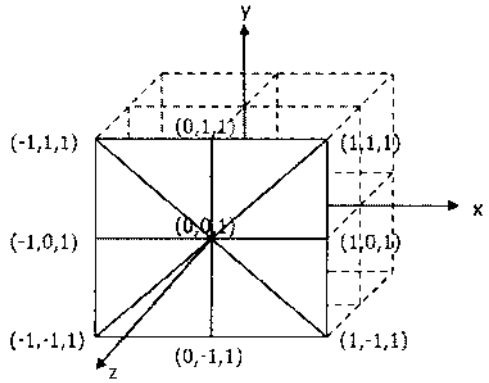

Figura 7 - Face referência para o desenho dos cubos

Serão desenhados oito triângulos por face, pelo que são necessários 9 vértices. Este número deve-se ao modelo de iluminação utilizado – "Modelo de Phong" – com sombreamento por interpolação, que favorece a qualidade do modelo com o aumento do número de subdivisões da face. A definição dos triângulos é feita por referência a um índice de uma lista de vértices (que devem ser definidos no sentido anti-horário).

Construção do modelo:

1. Rotação em torno de YY ou XX, para colocar a face na posição correcta do cubo a que pertence. Percorrendo todas as faces, o cubo é desenhado;
2. Transladar o cubo construído para a posição adequada do modelo usando os parâmetros de translação locais de cada cubo;
3. Rodar o cubo de acordo com o valor de rotação local (rotação de discos). Percorrendo todos os cubos, o modelo é desenhado;
4. Usando os parâmetros de rotação globais, rodar todo o modelo, segundo os três eixos, para os ângulos apropriados.

O resultado final, utilizando projecção perspectiva e efeitos de iluminação, é apresentado na figura 8.

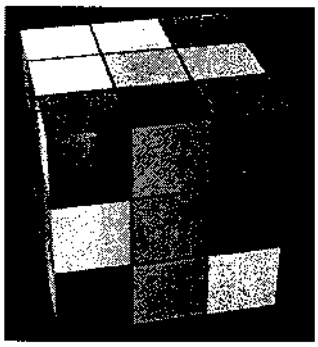

Figura 8 - O Cubo de Rubik

VII. MELHORAMENTOS FUTUROS

Pretende-se a curto/médio prazo introduzir alguns melhoramentos no que respeita à jogabilidade do Cubo de-Rubik:

- Manipulação dos discos do modelo por arrastamento do rato;
- Rotação global do modelo por arrastamento do rato;
- Incorporação de um algoritmo de auto-resolução do modelo, que permita dar uma ajuda valiosa aos iniciados nesta aventura [6].

O uso da capacidade de arrastamento do rato é uma forma de explorar as capacidades de movimento deste dispositivo, melhorando assim o nível de interacção do utilizador.

VIII. AGRADECIMENTOS

Agradece-se ao Prof. Joaquim Madeira as ideias para a realização do trabalho e o incentivo na escrita deste artigo.

REFERÊNCIAS

[1] Rubik's Cube History, http://cubeland.free.fr/infos/ernorubik.htm (Jan. 2005)

[2] Lecture Notes on the Mathematics of the Rubik's Cube, http://web.usna.navy.mil/~wdj/rubik_nts.htm (Jan. 2005)

[3] The OpenGL Oficial Site, http://www.opengl.org (Jan. 2005)

[4] E. Angel, "Interactive Computer Graphics: a top-down approach with OpenGL", 3rd ed., Addison-Wesley 2003, pp. 114–121

[5] Picking in OpenGL, http://www.soe.ucsc.edu/classes/cmps160/Fall01/picking.txt (Jan. 2005)

[6] Rubik's Cube Solution, http://www.nerdparadise.com/puzzles/333 (Jan. 2005)