Transformações Projetivas e Gráficos de Superfícies *

Lenimar N. Andrade

3 de julho de 1999

Resumo

Este texto descreve alguns tipos de transformações projetivas do $\mathbb{R}^3$ e alguns algoritmos que podem ser usados na construção do gráfico de uma superfície. É descrito também um simples formato de dados que pode ser usado na construção de diversos objetos em "modelos de arame" (*wireframe*).

Sumário

Projeções são transformações geométricas que, em geral, levam pontos de um espaço de dimensão n em pontos de um espaço de dimensão menor do que n. São fundamentais para a visualização de objetos 3D. Podem ser classificadas em duas famílias: as **paralelas** – onde o centro de projeção é considerado "no infinito" – e as **projetivas** (não-lineares) – com centro de projeção a uma distância finita dos pontos projetados. Algumas projeções paralelas são transformações lineares.

Veremos a seguir alguns exemplos de projeções.

*Disponível em ftp://mat.ufpb.br/pub/docs/cursos/proj.zip

1 Projeções ortográficas

São as transformções lineares P_x, P_y e P_z definidas pelas matrizes

$$[P_x] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad [P_y] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{e} \quad [P_z] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Logo, $P_x(x, y, z) = (0, y, z)$, $\quad P_y(x, y, z) = (x, 0, z) \quad$ e $\quad P_z(x, y, z) = (x, y, 0)$.

2 Projeções axonométricas

Consistem em duas rotações seguidas de uma projeção ortográfica. As duas rotações devem ser em torno de eixos distintos; a projeção deve ser na direção de um eixo diferente dos das rotações.

Por exemplo, fixados um ângulo de rotação θ em torno do eixo z e um ângulo de rotação ϕ em torno do eixo y, temos a seguinte matriz para uma projeção axonométrica T_x:

$$[T_x] = [P_x][R_\phi][R_\theta] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\operatorname{sen}\phi \\ 0 & 1 & 0 \\ \operatorname{sen}\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

portanto

$$[T_x] = \begin{bmatrix} 0 & 0 & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ \cos\theta\ \operatorname{sen}\phi & -\operatorname{sen}\theta\ \operatorname{sen}\phi & \cos\phi \end{bmatrix}$$

e daí temos

$$T_x(x, y, z) = (0,\ x\ \operatorname{sen}\theta + y\cos\theta,\ x\cos\theta\ \operatorname{sen}\phi - y\ \operatorname{sen}\theta\ \operatorname{sen}\phi + z\cos\phi).$$

Obtemos assim que a projeção por T_x do ponto (x, y, z) é o ponto do plano yOz :

$$(x\ \operatorname{sen}\theta + y\cos\theta,\ x\cos\theta\ \operatorname{sen}\phi - y\ \operatorname{sen}\theta\ \operatorname{sen}\phi + z\cos\phi).$$

De modo semelhante, podemos obter as matrizes das projeções axonométricas em outras direções.

3 Projeções oblíquas

São as projeções em que as retas que passam pelos pontos a serem projetados e suas respectivas projeções em um plano previamente escolhido não são perpendiculares ao plano de projeção.

Seja $\vec{v} = (a, b, c)$ um vetor em que $a \neq 0$. Escolhendo o plano de projeção como sendo o plano $x = 0$, vamos determinar a projeção (x', y', z') nesse plano de um ponto qualquer (x, y, z) do espaço na direção do vetor $\vec{v}$. Neste caso devemos ter que o vetor determinado pelos pontos (x, y, z) e (x', y', z') deve ser paralelo ao vetor $\vec{v}$, ou seja, existe um escalar k tal que $x' - x = ka$, $y' - y = kb$, $z' - z = kc$. Como (x', y', z') pertence ao plano de projeção, devemos ter $x' = 0$,

e daí $k = -x/a$, o que implica $y' = (-b/a)x + y$ e $z' = (-c/a)x + z$. Logo, a matriz desta projeção (em coordenadas cartesianas) é:

$$[T] = \begin{bmatrix} 0 & 0 & 0 \\ -\frac{b}{a} & 1 & 0 \\ -\frac{c}{a} & 0 & 1 \end{bmatrix}.$$

Projeções oblíquas com relação aos planos $y = 0$ ou $z = 0$ são feitas de forma análoga. Projeções com relação a outros planos podem ser feitas com auxílio de translações e rotações para que a projeção seja feita com relação a um dos planos coordenados.

4 Projeções perspectivas

As projeções perpectivas também são chamadas de *projeções cônicas*. Dados um plano α (plano de projeção) e um ponto $O \notin \alpha$ (posição do observador), a projeção perspectiva de um ponto $P \in \mathbb{R}^3$ é a interseção da reta que passa por O e P com o plano α.

Sejam $P(x, y, z)$ um ponto qualquer no espaço, $ax + by + cd + e = 0$ a equação de um plano α e $O(o_1, o_2, o_3)$ um ponto fora desse plano (veja figura 1). Queremos determinar a interseção da reta OP com α.

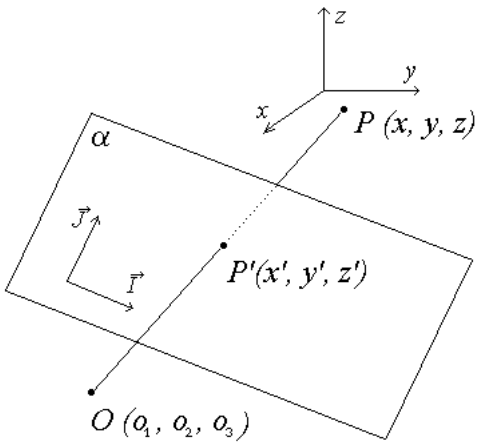

Figura 1: Projção perspectiva no plano α

A equação de OP é

$$\begin{cases} x' &= o_1 + t(x - o_1) \\ y' &= o_2 + t(y - o_2) \\ z' &= o_3 + t(z - o_3) \end{cases}, \ t \in \mathbb{R}.$$

Substitiuindo essas equações na equação de α obtemos

$$a[o_1 + t(x - o_1)] + b[o_2 + t(y - o_2)] + c[o_3 + t(z - o_3)] + d = 0$$

que é equivalente a

$$[ao_1 + bo_2 + co_3 + d] + t[a(x - o_1) + b(y - o_2) + c(z - o_3)] = 0$$

de onde obtemos o valor de t:

$$t = \frac{ao_1 + bo_2 + co_3 + d}{(ao_1 + bo_2 + co_3) - (ax + by + cz)}.$$

Este valor de t substituído na equação da reta nos fornece as coordenadas da interseção $P' = (\frac{A}{D}, \frac{B}{D}, \frac{C}{D})$, onde

$$\begin{aligned} A &= o_1(by + cz + d) - x(bo_2 + co_3 + d) \\ B &= o_2(ax + cz + d) - y(ao_1 + co_3 + d) \\ C &= o_3(ax + by + d) - z(ao_1 + bo_2 + d) \\ D &= (ax + by + cz) - (ao_1 + bo_2 + co_3) \end{aligned}$$

Daí, a transformação projetiva procurada não é uma transformação linear. Usando coordenadas homogêneas, temos que

$$T[x,\, y,\, z,\, 1] \,=\, [A/D,\, B/D,\, C/D,\, 1] \,=\, [A,\, B,\, C,\, D],$$

ou seja,

$$T[x,\, y,\, z,\, 1] \,=\, [o_1(by + cz + d) - x(bo_2 + co_3 + d),\;\; o_2(ax + cz + d) - y(ao_1 + co_3 + d),$$

$$o_3(ax+by+d) - z(ao_1 + bo_2 + d),\;\; (ax+by+cz) - (ao_1 + bo_2 + co_3)]$$

que pode ser escrito na forma

$$T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -bo_2 - co_3 - d & bo_1 & co_1 & do_1 \\ ao_2 & -ao_1 - co_3 - d & co_2 & do_2 \\ ao_3 & bo_3 & -ao_1 - bo_2 - d & do_3 \\ a & b & c & -ao_1 - bo_2 - co_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

É necessário que seja definido um sistema de coordenadas no plano α. Nos casos particulares em que α for paralelo a um dos planos coordenados ($x = 0$ ou $y = 0$ ou $z = 0$), podemos usar o sistema de coordenadas do plano coordenado ($\{\vec{j}, \vec{k}\}$ ou $\{\vec{i}, \vec{k}\}$ ou $\{\vec{i}, \vec{j}\}$) como se fosse o sistema de coordenadas de α.

No caso em que α for um plano qualquer, podemos determinar em α uma origem O e dois vetores unitários perpendiculares $\{\vec{I}, \vec{J}\}$. Assim, podemos escrever qualquer vetor $\overrightarrow{OP'}$ como combinação linear desses vetores: $\overrightarrow{OP'} = X\vec{I} + Y\vec{J}$ e, finalmente, desenhar na tela o ponto (X, Y) que representa as coordenadas de P' no plano α.

Por exemplo, a origem O de α e os vetores $\vec{I}$, $\vec{J}$ podem ser calculados da seguinte forma: sejam T a transformação projetiva anterior, $O = T[0, 0, 0, 1]$, $A = T[0, 1, 0, 1]$, $B = T[0, 0, 1, 1]$, $\vec{u} = \overrightarrow{OA} = A - O$ e $\vec{v} = \overrightarrow{OB} = B - O$, $\vec{w} = \vec{v} - \frac{\vec{u}\cdot\vec{v}}{\vec{u}\cdot\vec{u}}\vec{u}$. Então, $\vec{I} = \frac{\vec{u}}{||\vec{u}||}$ e $\vec{J} = \frac{\vec{w}}{||\vec{w}||}$ são ortonormais e, consequentemente, para calcular (X, Y) basta calcularmos os produtos internos $X = \overrightarrow{OP'} \cdot \vec{I}$ e $Y = \overrightarrow{OP'} \cdot \vec{J}$.

5 O *"formato 3DV"*

Um arquivo de dados no *"formato 3DV"* é um arquivo texto que contém uma lista de coordenadas de pontos no espaço 3D, seguida de uma lista de instruções de movimentos ou ligações entre esses pontos. Tem a seguinte estrutura:

N	Total de pontos
$x_1\ y_1\ z_1$	$1^{\underline{o}}$ ponto
$x_2\ y_2\ z_2$	$2^{\underline{o}}$ ponto
$\vdots$	$\vdots$
$x_N\ y_N\ z_N$	Último ponto
M	Total de movimentos e ligações
$p_1\ c_1$	$1^{\underline{o}}$ movimento ou ligação
$p_2\ c_2$	$2^{\underline{o}}$ movimento ou ligação
$\vdots$	$\vdots$
$p_M\ c_M$	Último movimento ou ligação

onde

- $(x_i,\ y_i,\ z_i)$ são as coordenadas do i-ésimo ponto no espaço 3D
- $(p_1,\ p_2,\ \cdots,\ p_M)$ é uma seqüência de inteiros positivos que define a ordem nos quais os pontos serão desenhados ou ligados. O primeiro ponto a ser desenhado ou ligado é o p_1-ésimo, o segundo é o p_2-ésimo, etc.
- $(c_1,\ c_2,\ \cdots,\ c_M)$ são as respectivas cores dos p_i-ésimos pontos. (dependendo da linguagem de programação, podem ser números inteiros de 0 a 15: 0 = preto, 1 = azul, 2 = verde, $\cdots$, 15 = branco).
- Se $c_i = 0$, então deve ser feito um movimento para o ponto p_i-ésimo. Se $c_i \neq 0$, então o ponto atual deve ser ligado com um segmento de reta ao ponto p_i-ésimo usando-se a cor c_i. Um movimento para um ponto significa um "pulo" para tal ponto, sem a necessidade de desenhá-lo.

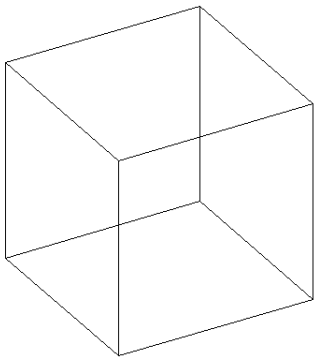

Figura 2: Cubo (posição 1)

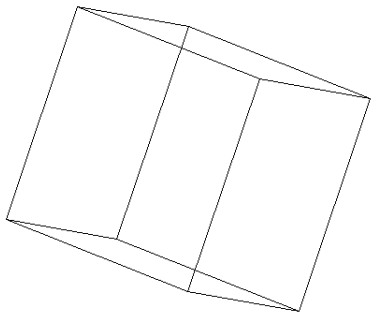

Figura 3: Cubo (posição 2)

3DV[1] é um pequeno programa (57 Kbytes) que deve ser executado com o computador em modo MS-DOS (não funciona no Windows 95) que pode ser usado para manipular objetos descritos nesse formato. Os objetos podem ser girados na tela e podem ser simples como um

[1]Disponível em ftp://ftp.unicamp.br/pub/simtelnet/msdos/graphics/3dv25.zip ou em ftp://mat.ufpb.br/pub/graph/3dv25.zip

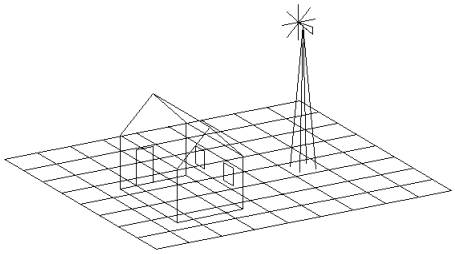

Figura 4: Casinha com moinho

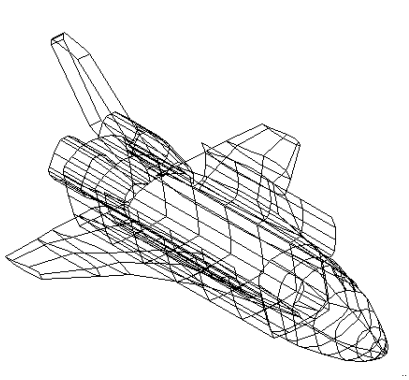

Figura 5: Avião

cubo (figura 2 ou 3), ou mais complexos como os da figura 4 ou figura 5. É necessária a ativação do *mouse* em modo MS-DOS.

Exemplo 5.1 *Um cubo (figura 2 ou 3) de cor 14 (amarelo em algumas implementações de linguagens de programação) com vértices* $(-1,-1,-1)$, $(-1,1,-1)$, $(1,1,-1)$, $(1,-1,-1)$, $(-1,-1,1)$, $(-1,1,1)$, $(1,1,1)$, $(1,-1,1)$ *pode ser descrito no* "formato 3DV" *da seguinte maneira:*

```
 8
-1   -1   -1
-1    1   -1
 1    1   -1
 1   -1   -1
-1   -1    1
-1    1    1
 1    1    1
 1   -1    1
 16
 1    0
 2   14
 3   14
 4   14
 1   14
 5   14
 6   14
 7   14
 8   14
 5   14
 2    0
 6   14
 3    0
 7   14
 4    0
 8   14
```

No cubo deste exemplo temos 8 pontos $(-1,-1,-1)$, $\cdots$, $(1,-1,1)$ *e 16 movimentos ou*

ligações. Inicialmente, é feito um movimento para o ponto 1, isto é, o ponto $(-1,-1,-1)$ *é considerado como sendo o primeiro ponto. Logo a seguir, o ponto 1 é ligado ao ponto 2 usando-se a cor 14. Dessa forma, é construída uma poligonal ligando-se os pontos* $1-2-3-4-1-5-6-7-8-5$ *com a cor 14. Finalmente, os pontos* $2-6$, $3-7$ *e* $4-8$ *são ligados.*

Depois que o programa tiver "entendido" o objeto descrito nesse formato, ele poderá movê-lo na tela usando as transformações já estudadas.

6 Gráficos de superfícies

Nesta seção descrevemos como construir o gráfico de uma superfície definida em forma paramétrica por $F(u,v) = (f_1(u,v),\ f_2(u,v),\ f_3(u,v))$, com $u \in [u_{\min}, u_{\max}]$ e $v \in [v_{\min}, v_{\max}]$.

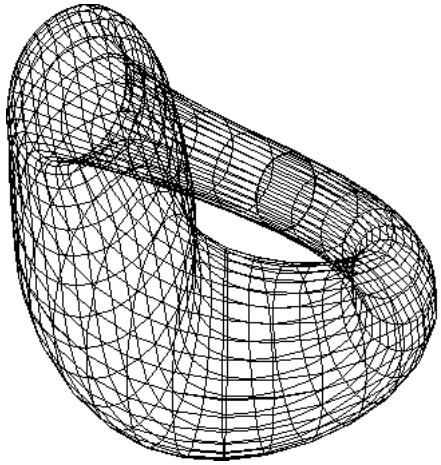

Figura 6: Garrafa de Klein

1 Inicialmente, escolhemos valores para as seguintes constantes:

- n_u: número de partes em que o intervalo $[u_{\min}, u_{\max}]$ é subdividido (Ex.: $n_u = 30$)
- n_v: número de partes em que o intervalo $[v_{\min}, v_{\max}]$ é subdividido (Ex.: $n_v = 25$)
- θ: ângulo de rotação em torno do eixo z, em radianos (Ex.: $\theta = 0.5$)
- ϕ: ângulo de rotação em torno do eixo y, em radianos (Ex.: $\phi = 1$)
- A, B: coordenadas do centro da tela (Ex.: $A = 400, B = 200$)
- k: constante de ampliação do gráfico (Ex.: $k = 50$)

2 Fazemos as variáveis u, v percorrerem o domínio da parametrização. Para isso, u é incrementado de um valor constante $\text{incr}_u = (u_{\max} - u_{\min})/n_u$ até atingir o valor máximo. Fazemos o mesmo com v usando $\text{incr}_v = (v_{\max} - v_{\min})/n_v$. Desse modo, u e v assumem todos os valores dos seguintes conjuntos

$$\{u_{\min} + \text{incr}_u,\ u_{\min} + 2\,\text{incr}_u,\ u_{\min} + 3\,\text{incr}_u,\ \cdots,\ u_{\max}\}$$

$$\{v_{\min} + \text{incr}_v,\ v_{\min} + 2\,\text{incr}_v,\ v_{\min} + 3\,\text{incr}_v,\ \cdots,\ v_{\max}\}$$

Quando u e v assumem todos os valores especificados nos conjuntos acima, os retângulos de vértices (u, v), $(u - \text{incr}_u, v)$, $(u, v - \text{incr}_v)$, $(u - \text{incr}_u, v - \text{incr}_v)$ abrangem todo o domínio da parametrização.

3 Para cada um dos retângulos mencionados no item anterior, calculamos o valor de F em cada vértice, ou seja, calculamos os pontos $(f_1(u,v),\ f_2(u,v),\ f_3(u,v))$, $(f_1(u - \text{incr}_u, v),\ f_2(u - \text{incr}_u, v),\ f_3(u - \text{incr}_u, v))$, $(f_1(u, v - \text{incr}_v),\ f_2(u, v - \text{incr}_v),\ f_3(u, v - \text{incr}_v))$, $(f_1(u - \text{incr}_u, v - \text{incr}_v),\ f_2(u - \text{incr}_u, v - \text{incr}_v),\ f_3(u - \text{incr}_u, v - \text{incr}_v))$.

4 Calculamos as projeções X e Y dos quatro pontos (x, y, z) calculados no item anterior. Entre outras opções, pode ser usada uma projeção axonométrica; neste caso, temos:

$$\begin{aligned} X &= x\ \text{sen}\,\theta + y\cos\theta \\ Y &= x\cos\theta\ \text{sen}\,\phi - y\ \text{sen}\,\theta\ \text{sen}\,\phi + z\cos\phi \end{aligned}$$

5 Desenhamos na tela o ponto (X, Y). Para aumentar o tamanho do graáfico e possivelmente melhorar a visualização, convém multiplicar X e Y por uma constante de ampliação k.

No sistema de coordenadas do vídeo, a ordenada y aumenta de cima para baixo. Por isso, uma multiplicação por -1 deve ser feita na direção y.

Como o sistema de coordenadas do vídeo tem sua origem no canto superior esquerdo da tela, uma translação para o ponto (A, B) deve ser feita. Para que o gráfico seja desenhada com origem no centro da tela, A deve ser igual à metade da resolução horizontal do vídeo e B ser a metade da resolução vertical.

Por esses motivos, ao invés de marcarmos o ponto (X, Y) na tela, é melhor marcarmos o ponto $(A, B) + k(X, -Y) = (A + kX, B - kY)$.

6 Cada grupo de 4 vértices projetados no item anterior são ligados por segmentos de reta, dando origem a pequenos quadriláteros no espaço 3D. Esses quadriláteros formam o gráfico da superfície – veja figura 6, onde usamos a seguinte parametrização para a superfície:

$$F(u,v) = \begin{cases} (6\cos u(1 + \ \text{sen}\,u) + 4(1 - \frac{\cos u}{2})\cos(v+\pi),\ 16\ \text{sen}\,u, \\ \qquad\qquad 4(1 - \frac{\cos u}{2})\ \text{sen}\,v), \quad \text{se}\ \ \pi < u \leq 2\,\pi; \\ (6\cos u(1 + \ \text{sen}\,u) + 4(1 - \frac{\cos u}{2})\cos v\cos u, \\ \qquad 16\ \text{sen}\,u + 4(1 - \frac{\cos u}{2})\ \text{sen}\,u\cos v,\ 4(1 - \frac{\cos u}{2})\ \text{sen}\,v), \quad \text{se}\ 0 \leq u \leq \pi \end{cases}$$

Na figura 9 também foi usado este algoritmo.

7 Se quisermos desenhar os eixos coordenados, então calculamos as projeções dos pontos $(0,0,0)$, $(1,0,0)$, $(0,1,0)$ e $(0,0,1)$. Suponhamos que essas projeções sejam respectivamente iguais a (o_1, o_2), (x_1, x_2), (y_1, y_2) e (z_1, z_2). Podemos considerar o eixo x como sendo o segmento de reta $(o_1, o_2) - (x_1, x_2)$, o eixo y como sendo $(o_1, o_2) - (y_1, z_2)$ e o eixo z como sendo $(o_1, o_2) - (z_1, z_2)$.

7 Eliminação de curvas invisíveis

Para eliminar as curvas "invisíveis" do gráfico de uma superfície, existem vários algoritmos que podem ser utilizados. Nesta seção apresentaremos brevemente dois deles:

- o algoritmo do pintor – de natureza geral, pode ser usado não só com superfícies, mas com cenas complexas envolvendo diversos objetos.
- o algoritmo do horizonte flutuante – usado só com gráficos de funções $z = f(x, y)$.

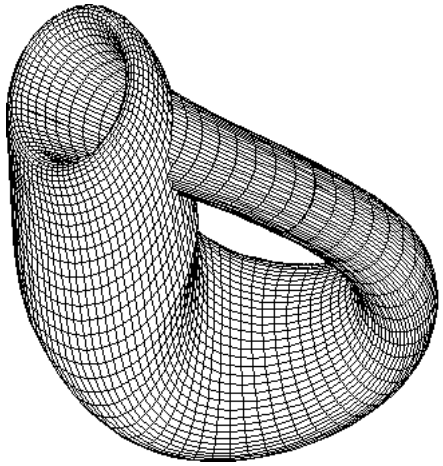

Figura 7: Garrafa de Klein

O algoritmo do pintor consiste em três etapas:

- cálculo de todos os quadriláteros que formam o gráfico da superfície (item 6 do algoritmo descrito na seção 6 anterior). Esse tipo de informação pode ser gravado em disco, por exemplo.
- classificação dos quadriláteros calculados no item anterior. Eles devem ser colocados em ordem decrescente segundo a distância do quadrilátero ao observador, isto é, eles devem ser dispostos de modo que apareçam primeiro os que estão mais distantes do observador e apareçam por último os que estão mais próximos. Podemos usar qualquer algoritmo de ordenação de conjuntos, por exemplo o algoritmo *quick sort*. Definimos a distância do observador a um quadrilàtero como sendo a distância entre o observador e um ponto do quadrilátero que pode ser, por exemplo, o centro do quadrilátero.
- colocamos na tela os quadriláteros ordenados no item anterior. Assim que um quadrilátero é mostrado na tela, ele é preenchido, ou seja, pintado com determinada cor.

Dessa forma, o algoritmo do pintor inclui, além dos algoritmos específicos de construção do gráfico um algoritmo de ordenação de conjuntos (conhecido dos cursos básicos de computação) e um algoritmo que seja conveniente para o preenchimento de regiões.

Veja na figura 7 a superfície mostrada na figura 6 após a aplicação desse algoritmo.

Podemos usar o vetor normal à superfície para pintá-la ou iluminá-la de várias formas. Podemos usar uma cor que dependa do ângulo que o vetor normal à superfície forma com o vetor normal ao plano de projeção. O vetor normal 'a superfície $F(u, v)$ no ponto (a, b) pode ser calculado pela expressão $\frac{\partial F}{\partial u}(a, b) \times \frac{\partial F}{\partial v}(a, b)$.

Podemos usar o algoritmo do horizonte flutuante (*floating horizon*) para desenhar gráficos de funções $z = f(x, y)$ (veja figura 8). Neste caso, o sentido em que o gráfico é construído é dos

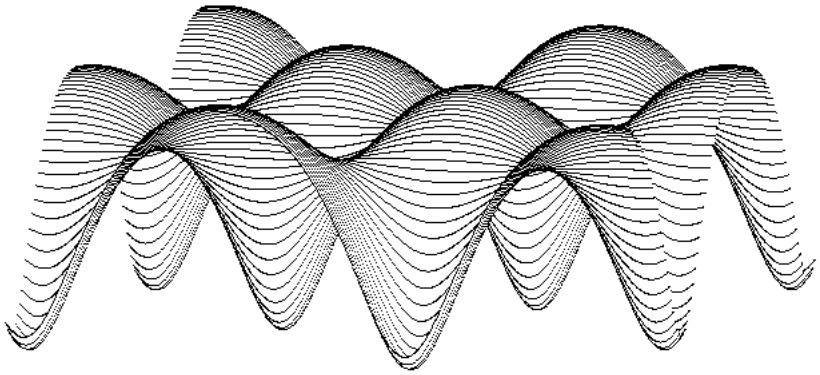

Figura 8: $z = 3\ \mathrm{sen}\, x\ \mathrm{sen}\, y/(2 +\ \mathrm{sen}\, x\ \mathrm{sen}\, y)$

pontos mais próximos para os mais distantes (ou seja, sentido inverso ao usado no algoritmo do pintor). Não é necessária ordenação de conjuntos pois, no caso do eixo x apontando para o observador, os pontos mais próximos são aqueles que tem maiores abscissas e os mais distantes são os que tem as menores abscissas.

Discretizamos o intervalo $[x_{\min}, x_{\max}]$. Obtemos assim um conjunto de pontos $\{x_{\min} = x_0, x_1, x_2, \cdots, x_{\max}\}$. Para cada x_i, construímos a curva $g_i(y) = f(x_i, y)$ observando o seguinte: um ponto do gráfico só será desenhado na tela se o valor de $g_i(y)$ for maior do que

$$\max(g_0(y),\ g_1(y),\ \cdots,\ g_{i-1}(y))$$

ou se for menor do que

$$\min(g_0(y),\ g_1(y),\ \cdots,\ g_{i-1}(y)).$$

O algoritmo do horizonte flutuante é descrito com mais detalhes em [2].

8 Pequeno programa em C

O gráfico da figura 9 foi construído pelo programa em C listado a seguir. Esse programa pode ser usado para construir o gráfico de muitas outras superfícies. É uma implementação do algoritmo descrito na seção 6.

```
/*          ================================================
            SUP3D.C - Construcao do grafico de uma superficie
            ================================================                */

#include <graphics.h>
#include <math.h>
#include <conio.h>

/* VARIAVEIS GLOBAIS */
```

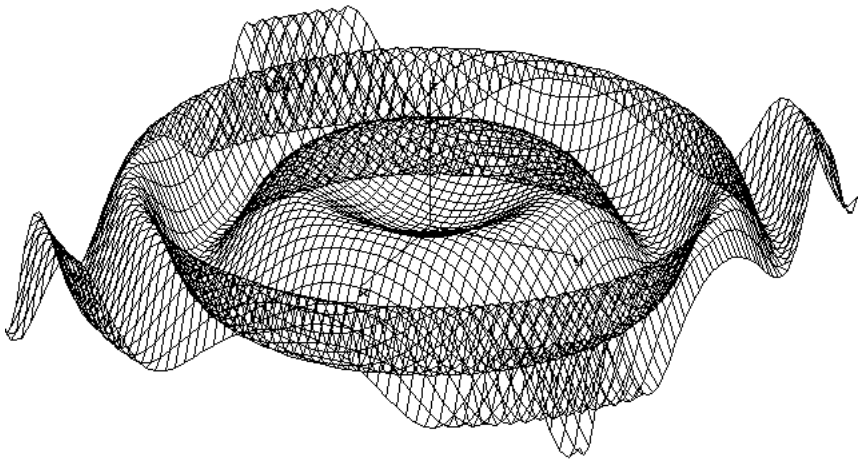

Figura 9: $F(u,v) = (u, v, \text{ sen } \frac{u^2+v^2}{4})$

```
float A, B,                 /* Coordenadas do centro da tela                */
      K = 40,               /* Coeficiente de ampliacao                     */
      nu = 60,              /* Quant. subdivisoes no dominio da variavel u  */
      nv = 60,              /* Quant. subdivisoes no dominio da variavel v  */
      fi = -M_PI/7,         /* Angulo de rotacao em torno do eixo y         */
      teta = -M_PI/7;       /* Angulo de rotacao em torno do eixo z         */

/* ------------------------------------------------------------------------ */

/* Definicao da parametrizacao   F(u,v) = (f1(u,v), f2(u,v), f3(u,v))
   da superficie cujo grafico e' desenhado. Neste caso, temos  a parametri-
                                                 2    2
   zacao da superficie   F(u, v) = (u, v, sen((u  + v )/4))                 */

float f1(float u, float v) { return u; }
float f2(float u, float v) { return v; }
float f3(float u, float v) { return sin((u*u + v*v)/4); }

/* ------------------------------------------------------------------------ */

int IniciaModoGrafico(void) {

/* Inicia o modo grafico.  O arquivo  .BGI associado ao tipo de video
   utilizado (Ex.: EGAVGA.BGI) precisa estar no subdiretorio atual ou
   ter  sua  localizacao  (Ex.: "c:\\utilit\\tc3\\bgi")  escrita como
   terceiro  parametro da funcao initgraph() . */

  int gm, gd = DETECT;
```

```
  initgraph(&gd, &gm, "");
  if (graphresult() != grOk) return 0;         /* Retorna o valor 0
                      se houver erro na mudanca para o modo grafico */

  return 1;    /* Retorna 1 se a operacao for realizada com sucesso */
}

/* ------------------------------------------------------------------------ */

void ProjetaPonto(float x, float y, float z, int *proj_X, int *proj_Y) {

/* Calcula as coordenadas do ponto (x, y, z) no plano de projecao. E' feita
   uma ampliacao de  K  unidades e uma translacao da origem  do  sistema de
   coordenadas do plano de projecao para o ponto (A, B). */

  float X, Y;

  /* Gira (x, y, z) de teta radianos em torno do eixo z e de fi radianos
     em torno do eixo y. E' feita uma projecao ortografica na direcao x. */

  X = y*cos(teta) + x*sin(teta);
  Y = x*cos(teta)*sin(fi) - y*sin(fi)*sin(teta) + z*cos(fi);

  /* Ampliacao e translacao de (X, Y) */
  *proj_X = (int) (A + K*X);
  *proj_Y = (int) (B - K*Y);
}

/* ------------------------------------------------------------------------ */

void DesenhaEixos(int k, int cor1, int cor2) {

/* Desenha os eixos x, y, z usando tamanho k e cores especificadas */

  int X0, Y0, X1, Y1, X2, Y2, X3, Y3;

  ProjetaPonto(0, 0, 0, &X0, &Y0); /* Calcula a projecao da origem (0, 0, 0) */

  setcolor(cor1);
  ProjetaPonto(k, 0, 0, &X1, &Y1);
  line(X0, Y0, X1, Y1);  /* Desenha o eixo x */
  setcolor(cor2);
  outtextxy(X1, Y1, "x");

  setcolor(cor1);
  ProjetaPonto(0, k, 0, &X2, &Y2);
```

```
  line(X0, Y0, X2, Y2);  /* Desenha o eixo y */
  setcolor(cor2);
  outtextxy(X2, Y2, "y");

  setcolor(cor1);
  ProjetaPonto(0, 0, k, &X3, &Y3);
  line(X0, Y0, X3, Y3);  /* Desenha o eixo z */
  setcolor(cor2);
  outtextxy(X3, Y3, "z");
}

/* ------------------------------------------------------------------------ */

void DesenhaSuperficie(float umin, float umax, float vmin, float vmax,
     int cor) {

/* Usando a  cor especificada, desenha a superficie definida pelas  equacoes
   parametricas (f1(u, v), f2(u, v), f3(u, v)) usando pequenos quadrilateros.
   E' usado como dominio o retangulo [umin, umax] x [vmin, vmax]. */

  float u, v, x, y, z, incrU, incrV;
  int X1, X2, X3, X4, Y1, Y2, Y3, Y4;

  incrU = (umax - umin)/nu;   /* Incremento da variavel u */
  incrV = (vmax - vmin)/nv;   /* Incremento da variavel v */

  setcolor(cor);

  u = umin + incrU;

  while (u < umax + incrU/2) {       /* Se nao houvesse erros de aproximacao
              esse "while" poderia ser substituido por "while (u <= umax)" */

    v = vmin + incrV;

    while (v < vmax + incrV/2) {     /* Se nao houvesse erros de aproximacao
              esse "while" poderia ser substituido por "while (v <= vmax)" */

      x = f1(u, v);
      y = f2(u, v);
      z = f3(u, v);
      ProjetaPonto(x, y, z, &X1, &Y1);

      x = f1(u - incrU, v);
      y = f2(u - incrU, v);
      z = f3(u - incrU, v);
```

```
      ProjetaPonto(x, y, z, &X2, &Y2);

      x = f1(u, v - incrV);
      y = f2(u, v - incrV);
      z = f3(u, v - incrV);
      ProjetaPonto(x, y, z, &X3, &Y3);

      x = f1(u - incrU, v - incrV);
      y = f2(u - incrU, v - incrV);
      z = f3(u - incrU, v - incrV);
      ProjetaPonto(x, y, z, &X4, &Y4);

      line (X1, Y1, X2, Y2);
      line (X4, Y4, X2, Y2);
      line (X3, Y3, X4, Y4);
      line (X3, Y3, X1, Y1);

      v = v + incrV;
    }          /* fim do "while (v < ... " */

    u = u + incrU;
  }            /* fim do "while (u < ... " */
}

/* ------------------------------------------------------------------------ */

void main(void) {

  if (!IniciaModoGrafico()) return;    /* Encerra o programa se houver erro
                                          na mudanca para o modo grafico */

  A = getmaxx()/2, B = getmaxy()/2;        /* Define  (A, B)  como sendo as
                                              coordenadas do centro da tela */
  DesenhaSuperficie(-6, 6, -6, 6, RED);
  DesenhaEixos(3, YELLOW, WHITE);

  getch();                          /* Espera ser pressionado qualquer tecla */
  closegraph();                     /* Encerra o modo grafico                */
}

/* ------------------------------------------------------------------------ */

/*** FIM DE "SUP3D.C" ***/
```

9 Exemplos de algumas parametrizações de superfícies

Nas equações mostradas a seguir R, a e b são constantes reais. Na maioria desses casos, os parâmetros u e v podem assumir valores no intervalo $[-\pi, \pi]$.

Superfície	*Parametrização*
Esfera de raio R	$F(u,v) = (R\cos u\ \mathrm{sen}\, v,\ R\cos u\cos v,\ R\ \mathrm{sen}\, u)$
Cilindro circular de raio R	$F(u,v) = (R\cos u,\ R\ \mathrm{sen}\, u,\ v)$
Cone	$F(u,v) = (u\cos v,\ u\ \mathrm{sen}\, v,\ u)$
Helicóide	$F(u,v) = (u\cos v,\ u\ \mathrm{sen}\, v,\ v)$
Toro	$F(u,v) = ((a+b\cos u)\cos v,\ (a+b\cos u)\ \mathrm{sen}\, v,\ b\ \mathrm{sen}\, u)$
Catenóide	$F(u,v) = (\cosh v\cos u,\ \cosh v\ \mathrm{sen}\, u,\ v)$
Hiperbolóide de 1 folha	$F(u,v) = (\cos u - v\ \mathrm{sen}\, u,\ \ \mathrm{sen}\, u + v\cos u,\ v)$
Hiperbolóide de 2 folhas	$F(u,v) = (\cos v\ \mathrm{senh}\, u,\ \ \mathrm{sen}\, v\ \mathrm{senh}\, u,\ \cosh u)$
Gráfico da função $z = f(x,y)$	$F(u,v) = (u,\ v,\ f(u,v))$

Referências

[1] D. Rogers, J. Adams, *Mathematical elements for Computer Graphics*, McGraw Hill, 1990.

[2] D. Rogers, *Procedural elements for Computer Graphics*, McGraw Hill, 1985.

[3] R. Plastock, G. Kalley, *Theory and problems of Computer Graphics*, Schaum's Outline Series, McGraw Hill, 1986.