IT-74

UNIVERSIDADE EDUARDO MONDLANE
FACULADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Trabalho De Licenciatura

TEMA : Programação Gráfica Tridimensional - Sua Aplicação No Desenho Da Superfície Z=F(X, Y)

Autor:

- **SÉRGIO LOURENÇO NUVUNGA**

Março, 2003

IT-74

IT-74

I.I.-24

UNIVERSIDADE EDUARDO MONDLANE
FACULADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Trabalho De Licenciatura

TEMA : Programação Gráfica Tridimensional - Sua Aplicação No Desenho Da Superfície Z=F(X, Y)

Supervisor:
- **Dr. NGUYEN NGOC QUYNH**

Março, 2003

D MATEMÁTICA U. E. M.
BIBLIOTECA
N.º 9995
DATA 20/10/2004
AQUISIÇÃO Oferta
CDU II-74

DEDICATÓRIA

Aos meus pais, Lourenço Manuel Nuvunga e Maria Lucas Chichava.

AGRADECIMENTOS

Estes agradecimentos vão para aquelas pessoas que me apoiaram quando precisei e que grandemente contribuíram para a realização deste trabalho.

O meu especial apreço vai para os professores **Dr. Nguyen Ngoc Quynh**(Supervisor) e **Dr. Manuel Alves** pela pronta disposição na resolução de questões pontuais que foram surgindo ao longo da realização do trabalho.

Quero, por último, agradecer aos funcionários da biblioteca do DMI que, directa ou indirectamente contribuíram para a realização deste trabalho. Muito obrigado.

DECLARAÇÃO DE HONRA

Declaro por minha honra, que este trabalho é o resultado da minha investigação, e que não foi submetido para outro grau que não seja o indicado, "Licenciatura em Informática", pela Universidade Eduardo Mondlane.

Maputo, Março de 2003

Sérgio Lourenço Nuvunga

(Sérgio Lourenço Nuvunga)

RESUMO

Nascida da associação do computador com as tecnologias associadas à televisão, a área da programação gráfica foi até há pouco tempo explorada apenas por quantos tinham acesso a grandes computadores e a dispositivos de visualização de preços elevado, necessários à geração de imagens gráficas. Agora, devido aos desenvolvimentos tecnológicos da microelectrónica, é possível encontrar microcomputadores, pequenos e a preços convidativos, suficientemente poderosos para suportar aplicações da programação gráfica de alto nível. Por isso, a experimentação com a programação gráfica está acessível a uma vasta audiência. Contudo, a dificuldade real na aprendizagem da programação gráfica reside na natureza complexa do próprio assunto**[Plastock e Gordon, 1991]**.

O presente trabalho visa a criação de um sistema informático capaz de desenhar as superfícies da forma Z=F(X,Y) e também a criação de uma biblioteca de funções e procedimentos para o modo gráfico do Borland Pascal.

O trabalho está dividido em três partes:

- **Programação Gráfica** – Nesta parte são descritos conceitos tais como: transformações gráficas tridimensionais e aplicações da programação gráfica.
- **Superfície Z=F(X,Y)** – Esta parte debruça-se sobre a definição de uma superfície, forma de geração, nomenclatura, classificação, ordem, exemplos de superfícies dadas na forma Z=F(X,Y) e descrição do algoritmo usado para o esboço do seu gráfico.
- **Implementação do algoritmo**(Horizonte Flutuante) – É nesta parte do trabalho onde o algoritmo é codificado numa linguagem concreta(Borland Pascal 7.0).

Os resultados obtidos são constituídos por uma aplicação que desenha as superfícies dadas na forma Z=F(X,Y) e várias unidades(bibliotecas) de funções e procedimentos que duma maneira combinada permitem uma melhor apresentação de informações no écrã, quando se está no modo gráfico do Borland Pascal.

INDICE

Página

INDICE DE FIGURAS

Página

1 INTRODUÇÃO

Ao longo dos séculos, artistas, engenheiros, projectistas, desenhadores, cartógrafos e arquitectos têm tentado resolver as dificuldades e restrições impostas pelo problema de representar um objecto ou uma cena tridimensional num meio bidimensional – o problema da *projecção.*

No passado, o cálculo e o desenho de gráficos de superfícies eram feitos manualmente; despendia-se muito tempo e não havia exactidão nos resultados obtidos, devido às possíveis falhas nos cálculos. Com o surgimento dos computadores, hoje em dia, esse trabalho é executado com uma margem ínfima de erro.

A capacidade de representar um ponto calculado permite-nos usar o ecrã como um instrumento de desenho. A descrição matemática do processo de projecção permite-nos a visualização de imagens de objectos tridimensionais em cenas que desejarmos. Os dois métodos básicos de projecção – *perspectiva* e *paralela* – foram criados para resolver o problema básico, mas mutuamente exclusivo, da representação pictórica: mostrar um objecto tal como o vemos e preservar a sua verdadeira grandeza e forma**[Plastock et al, 1991]**.

Com este trabalho pretende-se mostrar como a programação gráfica tridimensional pode ser usada para a criação de um sistema informático capaz de permitir o desenho de superfícies dadas na forma explícita da forma Z = F(X, Y).

Superfícies tridimensionais (espaciais) desempenham um papel importante:

- Na engenharia, projecto e manufactura de uma ampla gama de produtos, como automóveis, cascos de navios e asas de aviões, lâminas de propulsão, sapatos, garrafas, edificações, etc.
- Na descrição e interpretação de fenómenos físicos em áreas como geologia, física e medicina.

Neste trabalho foi dada mais ênfase à programação gráfica, visto que um dos objectivos é diminuir as dificuldades que alguns estudantes enfrentam no que se refere à este assunto, pelo que a superfície Z=F(X, Y) é apenas um caso de estudo para uma melhor compreensão da programação gráfica.

1.1 DEFINIÇÃO DO PROBLEMA

Macome (1995), citando **Lundberg**, afirma que, em sentido mais amplo, o problema da investigação é determinado pelas circunstâncias que despertam em alguém, curiosidade que reclama satisfação. Neste caso, a "Curiosidade", é em seu sentido mais geral. Na realidade, se o problema fosse exclusivamente determinado pela "curiosidade", do cientista, o progresso científico seria desordenado, individual e tendencioso, o que certamente é reconhecido por **Lundberg** quando diz que "a investigação científica é concebida ordinariamente como actividade de importância geral para a comunidade, antes de ser um esforço individual do estudioso para alcançar uma satisfação pessoal da curiosidade, embora esta última possa ser o principal dos incentivos do estudioso".

O presente trabalho não foge do referido no parágrafo anterior, pois procura combinar a curiosidade do autor em desenvolver uma aplicação em Turbo Pascal, capaz de desenhar gráficos de superfícies da forma Z=F(X,Y), com a necessidade de se diminuírem as dificuldades de alguns estudantes, no que se refere à programação gráfica. A maior parte dos estudantes do DMI, programa bem em mais do que uma linguagem, mas são poucos os que dominam a programação gráfica. Assim sendo, foram desenvolvidas algumas bibliotecas de funções e procedimentos que facilitarão sobremaneira o trabalho destes com a unidade gráfica do Turbo Pascal. Procurou-se também estudar alguns conceitos que caracterizam as superfícies no geral, bem como, investigar um algoritmo capaz de desenhar as superfícies da forma Z=F(X,Y).

2 OBJECTIVOS

2.1 OBJECTIVOS GERAIS

- Criar um sistema informáticc capaz de desenhar as superfícies do tipo Z = F(X, Y) num meio computacional;
- Criar uma biblioteca de funções e procedimentos da unidade gráfica do Turbo Pascal.

2.2 OBJECTIVOS ESPECÍFICOS

- Investigar as técnicas usadas na programação gráfica;
- Descrever as propriedades, definições, nomenclaturas e classificações da superfície Z= F(X, Y);
- Desenvolver um algoritmo para o desenho de superfícies do tipo Z = F(X, Y);
- Codificar o algoritmo desenvolvido num meio computacional.

3 MATERIAL E MÉTODOS

Durante a realização deste trabalho foram usadas as seguintes metodologias:

- Metodologia Descritiva, que consistiu na abordagem detalhada dos elementos que serviram de base para o alcance dos objectivos acima definidos, descrevendo as técnicas de programação gráfica , as propriedades, definições, nomenclaturas e classificações da superfície Z = F(X, Y) e o algoritmo que foi usado para a sua representação gráfica;
- Revisão Bibliográfica através da navegação à Internet, consulta aos livros e brochuras existentes relacionados com o tema;
- Consultas permanentes ao supervisor e outros especialistas em computação gráfica e geometria.
- Para a documentação do trabalho foi usado o Microsoft Word 97 e para a programação da aplicação que desenha as superfícies foi usado o Borland Pascal 7.0.

4 PROGRAMAÇÃO GRÁFICA

4.1 TRANSFORMAÇÕES GRÁFICAS TRIDIMENSIONAIS

4.1.1 INTRODUÇÃO

A manipulação, a visualização e a construção de imagens gráficas tridimensionais requer a utilização de transformações de coordenadas e de transformações geométricas tridimensionais. Estas transformações são formadas pela composição das transformações básicas de translação, de variação de escala e de rotação. Cada uma destas transformações pode ser representada por uma matriz. Isto permite construir transformações mais complexas através da multiplicação ou concatenação de matrizes.

Tal como nas transformações bidimensionais, são adoptados dois pontos de vista complementares: O objecto (ou imagem) é manipulado directamente através da utilização de transformações geométricas ou o objecto permanece fixo e o sistema coordenado do observador é deslocado pela utilização de transformações de coordenadas. Além disso, a construção de objectos (ou imagens) complexos é facilitada pela utilização de transformações de instanciação, as quais combinam ambos pontos de vista**[Plastock et al, 1991]**.

4.1.2 TRANSFORMAÇÕES GEOMÉTRICAS

Em relação a um sistema coordenado tridimensional, um objecto *Obj* é considerado como um conjunto de pontos:

$$Obj = \{P(x, y, z)\}$$

Se o objecto é movido para uma nova posição, podemos considerá-lo como um novo objecto *Obj'*, no qual todos os pontos coordenados P'(x', y', z') podem ser obtidos a partir dos pontos coordenados originais P(x, y, z) de *Obj* através da aplicação de uma transformação geométrica**[Plastock et al, 1991]**.

Translação

Um objecto é deslocado uma dada distância e segundo uma dada direcção a partir da sua posição inicial. A direcção e o deslocamento da translação é determinada pelo vector:

$$V = aI + bJ + cK$$

Onde **I, J** e **K** são os vectores unitários do espaço vectorial R^3.

As coordenadas de um ponto transladado podem ser calculadas usando a transformação:

$$T_v = \begin{cases} x' = x + a \\ y' = y + b \\ z' = z + c \end{cases}$$

(Ver Fig. 4-1). Para representar esta transformação através de uma matriz precisamos de usar coordenadas homogéneas[1]. A respectiva transformação na forma matricial homogénea, pode então ser expressa da forma seguinte:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

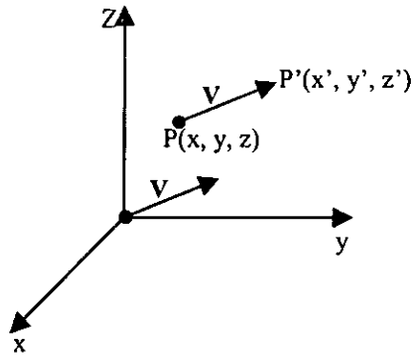


Fig.4-1 Translação.

[1] As coordenadas homogéneas permitem que a transformação de translação e que a transformação da projecção perspectiva sejam representadas por matrizes.

Variação de escala

O processo de variação de escala altera as dimensões de um objecto. O factor de escala *s* determina se a variação de escala é uma ampliação, $s > 1$, ou uma redução, $s < 1$.

A variação de escala em relação à origem é efectuada pela transformação:

$$S_{s_x, s_y, s_z} : \begin{cases} x' = S_x \cdot x \\ y' = S_y \cdot y \\ z' = S_z \cdot z \end{cases}$$

Na forma matricial temos:

$$S_{s_x, s_y, s_z} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix}$$

Rotação

A rotação em três dimensões é consideravelmente mais complexa que a rotação em duas dimensões. Em duas dimensões a rotação é determinada por um ângulo de rotação θ e um centro de rotação P. As rotações tridimensionais requerem a definição de um centro de rotação e de um eixo de rotação. Uma rotação é chamada *canónica* quando algum dos eixos de coordenadas, O_x, O_y ou O_z, é escolhido como eixo de rotação. Então a transformação de rotação processa-se tal como no caso da rotação bidimensional em torno da origem(Ver Fig. 4-2).

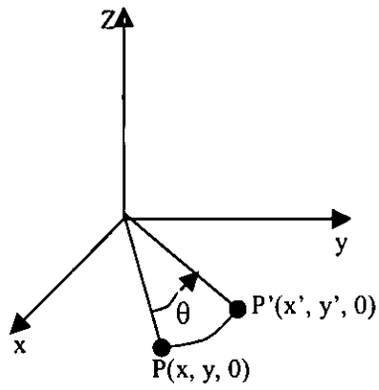


Fig. 4-2 Rotação.

Rotação em torno do eixo O_z

$$R_{\theta,K}:\begin{cases} x' = x\cos\theta - y\,sen\,\theta \\ y' = x\,sen\,\theta + y\cos\theta \\ z' = z \end{cases}$$

Rotação em torno do eixo O_y

Uma dedução análoga conduz a:

$$R_{\theta,J}:\begin{cases} x' = x\cos\theta + z\,sen\,\theta \\ y' = y \\ z' = -x\,sen\,\theta + z\cos\theta \end{cases}$$

Rotação em torno do eixo O_x

Do mesmo modo:

$$R_{\theta,I}:\begin{cases} x' = x \\ y' = y\cos\theta - z\,sen\,\theta \\ z' = y\,sen\,\theta + z\cos\theta \end{cases}$$

As matrizes de transformação correspondentes são:

$$R_{\theta,K} = \begin{pmatrix} \cos\theta & -sen\,\theta & 0 \\ sen\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{\theta,K} = \begin{pmatrix} \cos\theta & 0 & sen\,\theta \\ 0 & 1 & 0 \\ -sen\,\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_{\theta,K} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -sen\,\theta \\ 0 & sen\,\theta & \cos\theta \end{pmatrix}$$

O caso geral de rotação em torno do eixo L pode ser construído a partir das rotações canónicas, usando a multiplicação de matrizes.

4.1.3 TRANSFORMAÇÕES DE COORDENADAS

Segundo **Plastock et al (1991)** Podemos também obter os efeitos de translação, de variação de escala e de rotação movendo o observador (que vê um objecto) em relação à um objecto fixo. Este tipo de transformação é denominado *transformação de coordenadas.* Primeiro, ligamos um sistema coordenado ao observador; depois movê-lo concomitantemente com o sistema coordenado que lhe está associado. De seguida, recalculamos as coordenadas do objecto observado em relação ao novo sistema coordenado do observador. O valor das novas coordenadas serão exactamente os mesmos que seriam se o observador tivesse permanecido fixo e o objecto se tivesse movido. O que corresponde a uma transformação geométrica(Fig. 4-3).

Se o deslocamento do sistema coordenado do observador, para uma nova posição, é determinado pelo vector V = *a*I + *b*J + *c*K, um ponto P(x, y, z) referido ao sistema coordenado original tem as coordenadas P(x', y', z') no novo sistema coordenado, e

$$\overline{T}_v : \begin{cases} x' = x - a \\ y' = y - b \\ z' = z - c \end{cases}$$

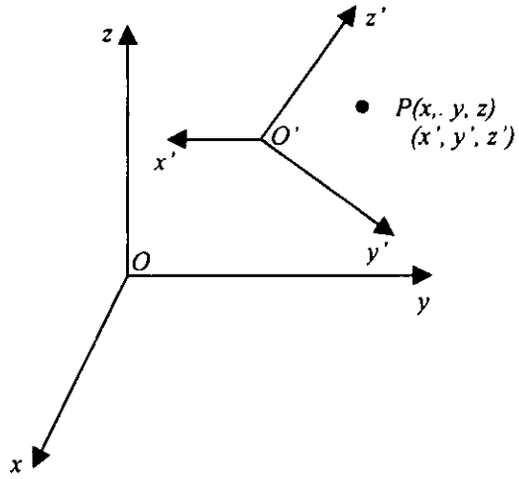


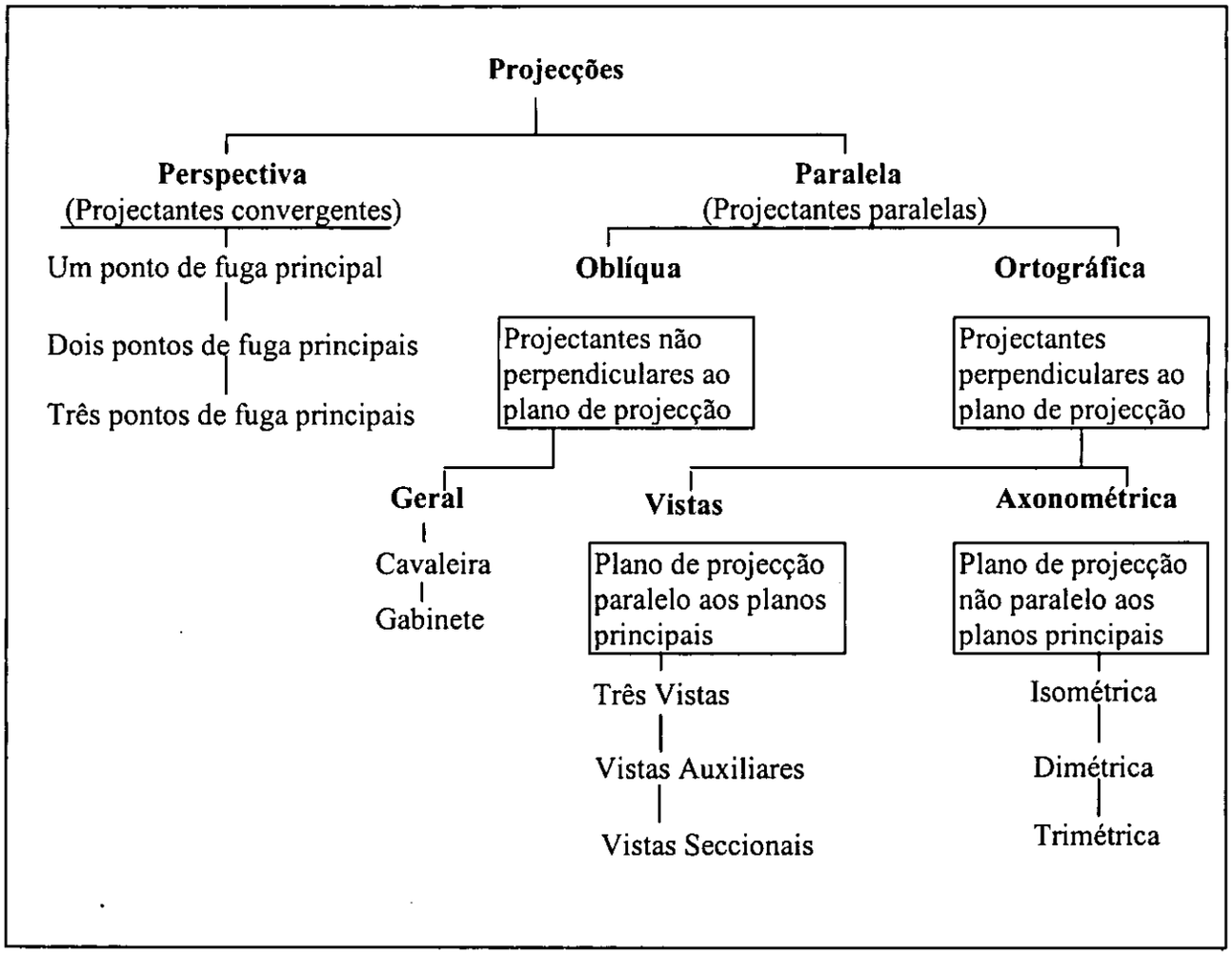
Fig. 4-3 Transformação de Coordenadas.

Determinações similares são válidas para as transformações de coordenadas relativas à variação de escala e à rotação.

4.2 OS DIFERENTES TIPOS DE PROJECÇÃO

Podemos construir diferentes tipos de projecção segundo a visão que se pretende obter do objecto. A tabela 4-1 fornece uma taxonomia das famílias de projecções perspectiva e paralela. Algumas têm nomes – cavaleira, gabinete, isométrica, etc. Outras projecções qualificam o tipo principal de projecção – perspectiva com um ponto de fuga principal, e por aí fora**[Plastock et al, 1991]**.

Tabela 4-1 Tipos de Projecções



4.2.1 PROJECÇÃO PERSPECTIVA

4.2.1.1 PRINCÍPIOS BÁSICOS

As técnicas de projecção perspectiva são generalizações dos princípios usados pelos artistas quando preparam desenhos em perspectiva de objectos ou cenas tridimensionais. O olho do artista coloca-se no *centro de projecção*, e o quadro, ou mais precisamente o plano que contém o quadro,

transforma-se no *plano de projecção*. Um ponto imagem está localizado na intersecção de uma *projectante* (Uma recta traçada pelo ponto objecto e pelo centro de projecção) com o plano de projecção (Fig. 4-4).

Os desenhos em perspectivas são caracterizados pelo encurtamento perspectivo e pelos pontos de fuga. O *encurtamento perspectivo* é a ilusão de que os objectos e comprimentos são cada vez menores à medida que a sua distância ao centro de projecção aumenta. A ilusão de que certos feixes de rectas paralelas se encontram num ponto é outra característica dos desenhos em perspectiva. A estes pontos dá-se o nome de *pontos de fuga*. Os *pontos de fuga* são formados pela aparente intersecção das rectas paralelas com um dos três eixos principais. O número de pontos de fuga principais é determinado pelo número de eixos principais intersectados pelo plano de projecção**[Plastock et al, 1991]**.

4.2.1.2 DESCRIÇÃO MATEMÁTICA DE UMA PERSPECTIVA

Uma transformação perspectiva é definida indicando um centro de projecção e um plano de projecção. O plano de projecção é definido pelo seu *ponto de referência* R_o e pela *normal* N ao plano de projecção. O *ponto do objecto* P está localizado em coordenadas universais em (x, y, z). O problema é determinar as coordenadas do *ponto da projecção* P(x', y', z') (Fig. 4-4).

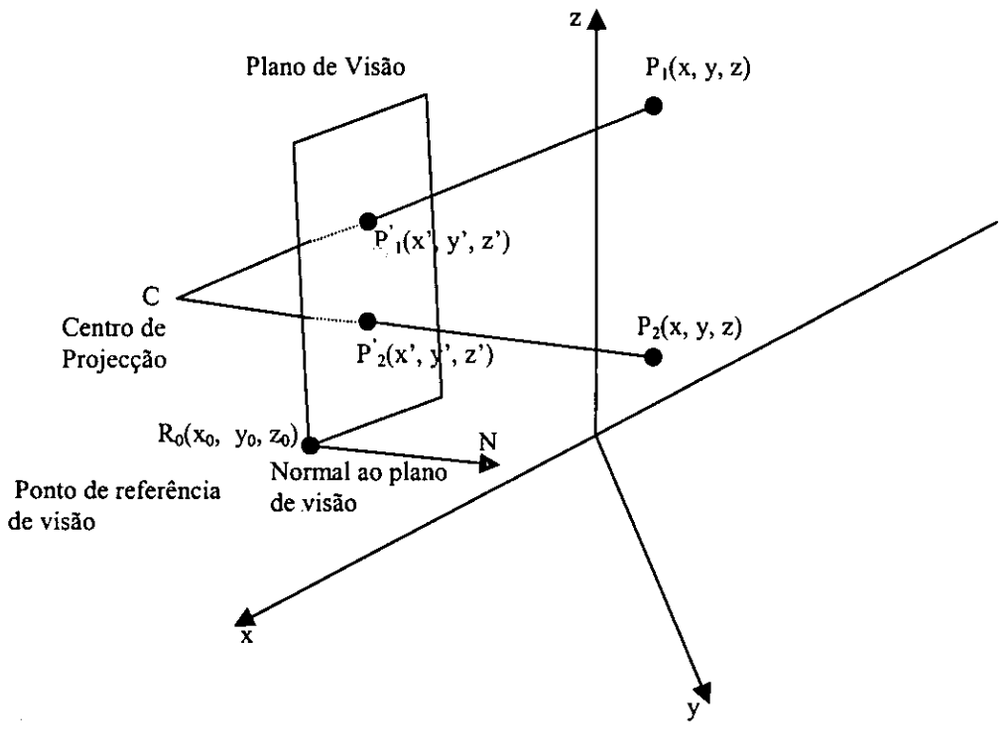


Fig. 4-4 Perspectiva.

Consideremos o seguinte exemplo da Fig. 4-5: Aí, o plano de projecção é o plano xOy, e o centro de projecção é o ponto C(0, 0, -d) na parte negativa do eixo Oz. Usando a semelhança dos triângulos ABC e A'OC, achamos:

$$x' = \frac{d \,.\, x}{z + d} \qquad y' = \frac{d \,.\, y}{z + d} \qquad z' = 0$$

A transformação perspectiva entre ponto do objecto e ponto da projecção não é linear e por isso não pode ser representada por uma matriz de transformação 3 x 3. Contudo, se usarmos coordenadas homogêneas, a transformação perspectiva pode ser representada por uma matriz 4 x 4:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} d \,.\, x \\ d \,.\, y \\ 0 \\ z + d \end{pmatrix} = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

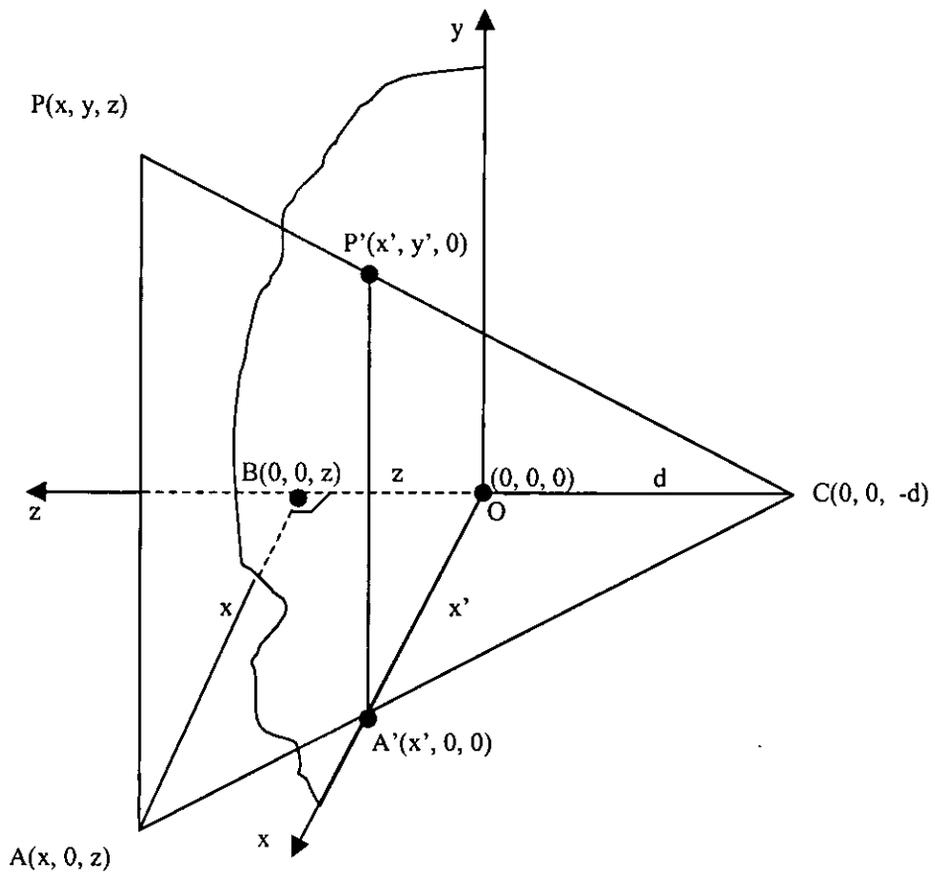


Fig. 4-5 Exemplo de Uma Perspectiva

4.2.1.3 ANOMALIAS DA PERSPECTIVA

O processo de construção de uma perspectiva introduz certas anomalias que aumentam o realismo em termos de profundidade, mas também alteram as medidas e formas reais.

- *Encurtamento perspectivo.* Quanto mais distante um objecto está do centro de projecção, menor parece ser (isto é, o tamanho da sua projecção torna-se menor).
- *Pontos de Fuga.* As projecções de rectas não paralelas ao plano de projecção (isto é, rectas que não são perpendiculares à normal ao plano de projecção), possuem algumas anomalias, uma manifestação comum de algumas dessas anomalias é a ilusão de que os carris do caminho-de-ferro se encontram num ponto do horizonte**[Plastock et al, 1991]**.

4.2.2 PROJECÇÃO PARALELA

4.2.2.1 PRINCÍPIOS BÁSICOS

Os métodos da projecção paralela são usados pelos desenhadores e engenheiros para criar desenhos, que são instrumento de trabalho, de um objecto, dos quais preservam a sua escala e forma. A completa representação destes pormenores requer muitas vezes duas ou mais vistas (Projecções) do objecto em diferentes planos de projecção.

Na projecção paralela, os pontos imagem encontram-se na intersecção de uma projectante, traçada pelo ponto do objecto e com uma direcção dada, com o plano de projecção (Fig. 4-6). A *direcção da projecção* é a mesma para todas as projectantes. As *projecções ortogonais* são caracterizadas pelo facto de a direcção de projecção ser perpendicular ao plano de projecção. Quando a direcção e projecção é paralela à algum dos eixos principais, podemos obter a vista de frente, a planta, e os alçados laterais usados nos desenhos de mecânica (também chamados de *desenhos de vistas*). As *projecções axonométricas* são projecções ortogonais em que a direcção de projecção não é paralela à nenhum dos três eixos principais. As projecções não ortogonais chamam-se *projecções paralelas obliquas***[Plastock et al, 1991]**.

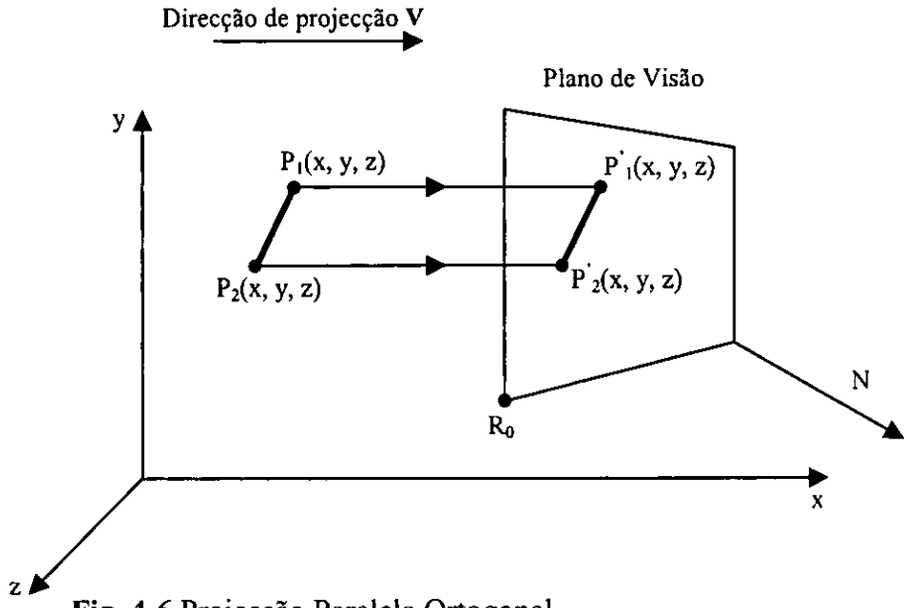


Fig. 4-6 Projecção Paralela Ortogonal.

4.2.2.2 DESCRIÇÃO MATEMÁTICA DE UMA PROJECÇÃO PARALELA

Plastock et al (1991), afirmam que uma *transformação projectiva paralela* é definida fixando a *direcção do vector de projecção* V e o plano de referência R_0 e pela normal ao plano de projecção N. O ponto do objecto P está localizado em (x, y, z) em coordenadas universais. O problema está em determinar as coordenadas do ponto de projecção P'(x', y', z') (ver Fig. 4-6).

Se o vector de projecção V tem a direcção da normal ao plano de projecção N, a projecção diz-se *ortogonal*. Caso contrário diz-se *oblíqua* (ver Fig. 4-7).

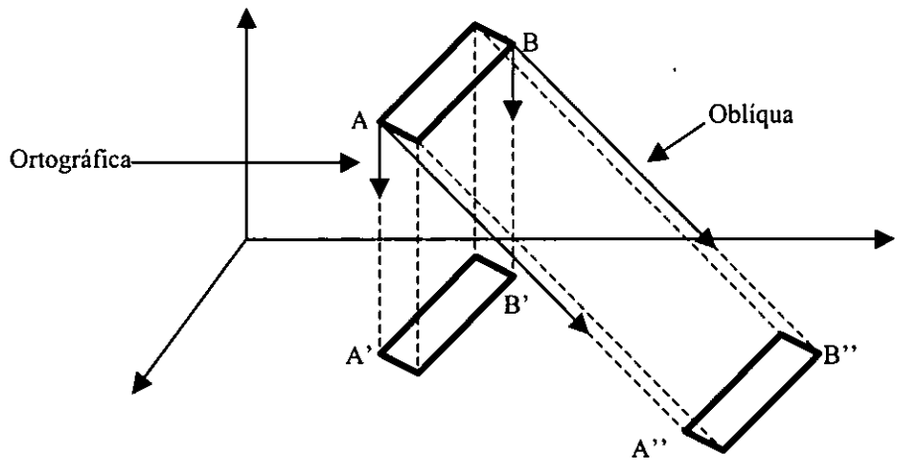


Fig. 4-7 Projecção Paralela Oblíqua.

Algumas subcategorias comuns das projecções ortogonais são:

- *Isométrica* – a direcção de projecção faz ângulos iguais com os três eixos principais.
- *Dimétrica* – a direcção de projecção faz ângulos iguais com dois dos eixos principais.
- *Trimétrica* – a direcção de projecção faz ângulos desiguais com os três eixos principais.

Algumas subcategorias comuns das projecções oblíquas são:

- *Cavaleira* – a direcção de projecção é escolhida de modo a que não haja encurtamento dos segmentos de recta perpendiculares ao plano xOy.
- *Gabinete* – a direcção de projecção é escolhida de modo a que os segmentos de recta perpendiculares ao plano xOy sejam encurtados de metade dos seus comprimentos.

4.3 BGI (BORLAND GRAPHICS INTERFACE)

4.3.1 DEFINIÇÃO

É um conjunto de instrumentos poderosos e fáceis de usar para a criação de programas gráficos, o BGI está disponível a partir da versão 4.0 do Turbo Pascal**[Weiskamp et al, 1989]**.

4.3.2 UNIDADE GRAPH DO TURBO PASCAL

A unidade graph representa decididamente a mais complexa de todas as unidades fornecidas pela Borland. Ela contém mais de 50 procedimentos, cerca de 20 funções, e resultados de constantes, variáveis e tipos de dados. Além disso a Borland inclui drivers para todas as placas gráficas importantes e fontes para desenhar uma ampla variedade de estilos de caracteres**[O'brien, 1992]**.

4.3.3 ESTRUTURA DE UM PROGRAMA BÁSICO BGI

O seguinte programa é provavelmente um dos mais curtos programas gráficos que pode ser escrito com o Borland Pascal. Ele mostra a estrutura básica de um programa gráfico do Borland Pascal, e pode ser usado como um modelo para a escrita de programas gráficos**[Weiskamp et al, 1989]**.

```
Program SmallestGraphicsProgram;
Uses
    Graph; {Unidade gráfica do Borland Pascal}
Const
    Gdriver : Integer = Detect; {Seleccionar o instrumento de autodetecção}
Begin
    InitGraph(Gdriver, Gmode, ''); {Inicializar o driver e o modo gráfico}
    {... Comandos para o desenho nesta área...}
    CloseGraph {Fechar o modo gráfico}
End.
```

4.3.4 MAPEAMENTO ENTRE COORDENADAS DO ECRÃ E UNIVERSAIS

No geral, deve ser possível fazer um mapeamento entre as coordenadas universais e as do ecrã, tal como está mostrado na Fig.4-8. Para tal, podem ser usadas as relações mostradas na Fig 4-9. As equações:

$$X' = a*X + b$$
$$Y' = c*Y + d$$

Definem como um ponto (X,Y) em coordenadas universais pode ser mapeado em coordenadas do ecrã (X',Y'). Os valores L1, T1, R1 e B1 representam o intervalo de valores que estão sendo considerados em coordenadas universais, similarmente, L2, T2, R2 e B2 definem o tamanho da área do ecrã na qual o objecto será mapeado**[Weiskamp et al, 1989]**.

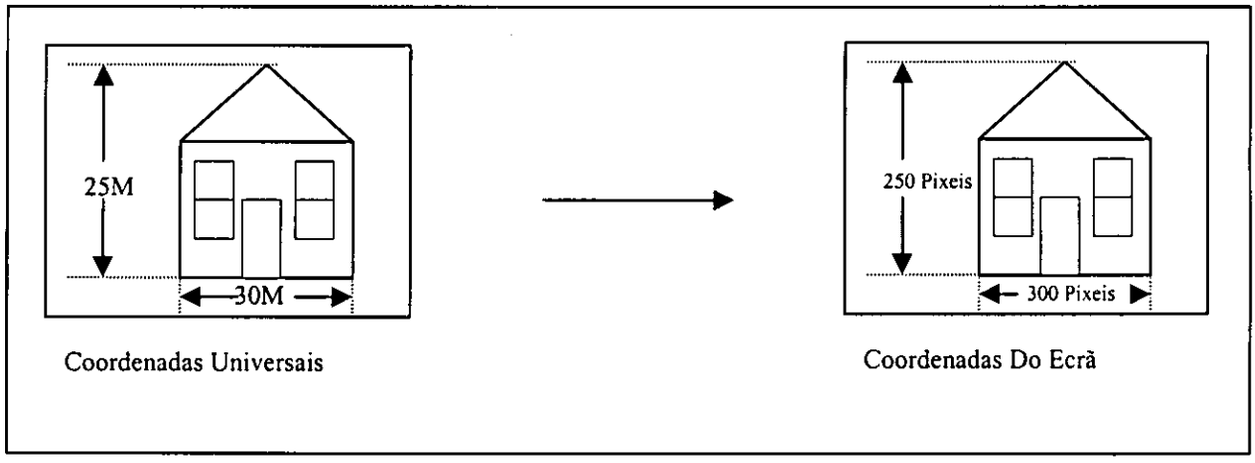


Fig. 4-8 Mapeamento De Objectos De Coordenadas Universais Para As Coordenadas Do Ecrã.

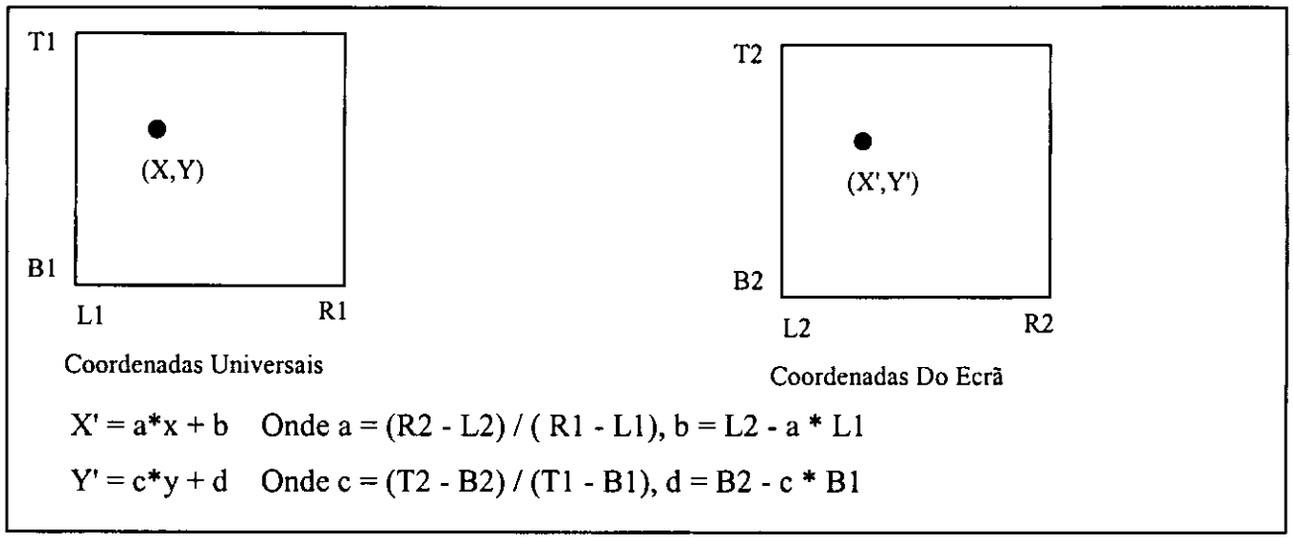


Fig. 4-9 Equações para o mapeamento das coordenadas, de universais para as do ecrã.

4.4 APLICAÇÕES DA PROGRAMAÇÃO GRÁFICA

4.4.1 INTRODUÇÃO

Para lá de um conhecimento básico dos conceitos da programação gráfica, o desenvolvimento de uma verdadeira aplicação programação gráfica requer conhecimentos em três áreas: (1) ergonomia do computador, (2) estruturas de informação, e (3) o campo de aplicação. Sem uma cuidadosa atenção à estas três áreas, as mais sofisticadas rotinas gráficas podem tornar-se inúteis**[Plastock et al, 1991]**.

4.4.2 ERGONOMIA DO COMPUTADOR

A *ergonomia do computador* diz respeito à interface homem-máquina (frequentemente chamada a *interface com o utilizador*). É muito importante uma vez que no final alguém tem de ser capaz de usar o sistema gráfico para produzir resultados úteis e significativos. O objectivo do responsável pelo desenvolvimento do sistema é atingir o equilíbrio entre o custo do tempo gasto a desenvolver, para o sistema, uma interface agradável com utilizador e o custo dos erros cometidos pelo utilizador. De um modo geral, são consideradas seis áreas no desenvolvimento de uma interface com o utilizador ergonomicamente correcta:

- Compatibilidade;
- Brevidade;
- Flexibilidade;
- Tempo de resposta;
- Consistência e
- Carga de trabalho do operador.

Compatibilidade

O termo <<compatibilidade>> significa que a esperada entrada de dados do utilizador e a saída do computador serão consistentes com o modelo que o utilizador tem do universo. Por exemplo, STOP escrito a vermelho será compatível com o utilizador, enquanto que STOP escrito a verde provocará confusão ao utilizador. Um sistema compatível reduz o volume de recodificação de informação requerida pelo utilizador. Uma boa compatibilidade pode ser obtida quando o utilizador é

confrontado com informação não conflituosa, pelo reforço das espectativas normais, que têm a ver com a relação entre a saída e a entrada esperada do utilizador, e pela apresentação da saída numa forma que permita o seu uso directo. Os símbolos gráficos chamado ícones podem ser usados para representar selecções no menu. Estes são exemplos vulgares da aplicação do conceito de compatibilidade**[Plastock et al, 1991]**.

Brevidade

Tanto a teoria psicológica como a experiência comum sugerem que há um limite superior para a quantidade de informação que a mente humana consegue processar num dado período de tempo. A quantidade de informação que uma pessoa normal consegue reter em memorização de curta duração está limitada a cerca de 7 ou 8 *bits*. (Aqui, um bit representa um agrupamento básico de informação.) por exemplo, dificilmente um número com sete dígitos (por exemplo, 1735824) consegue ser retido em memorização de curta duração.

Neste caso, considera-se que cada dígito é um bit. A mente humana tem sempre a tendência de agrupar itens e desse modo reduzir o número de bits de informação que têm que ser recordados. Por exemplo, o número 217-582-1894 pode ser de imediato recordado por grande parte das pessoas devido ao processo de *associação* que se verifica. A associação diz respeito ao agrupamento automático de informação pelo cérebro.

A brevidade é importante porque, apesar da associação, mensagens que contenham mais do que 8 bits de informação são normalmente difíceis de recordar e utilizar. Além disso, a informação deverá ser apresentada de um modo que conduza a associação. Isto consegue-se dando à informação semelhante a mesma localização no ecrã**[Plastock et al, 1991]**.

Flexibilidade

Um bom sistema pode adaptar-se ao nível de experiência de um utilizador. Por exemplo, os utilizadores inexperientes normalmente sentem-se mais à vontade com sistemas que usam diálogo selectivo (escolhas em menus). Contudo, depois do utilizador se tornar mais conhecedor do sistema, uma linguagem de comandos é preferível, porque o sistema de menus parece lento ao utilizador. Um bom sistema permitirá uma suave transição de um tipo de diálogo para outro**[Plastock et al, 1991]**.

Tempo De Resposta

Um sistema gráfico está normalmente instalado num local onde os utilizadores estão empenhados em actividades que requerem um elevado grau de concentração e de criatividade. Num tal ambiente, mesmo profissionais de computação sazonais tem tendência a esquecer rapidamente a complexidade das tarefas que o computador está a desempenhar e impacientam-se logo após alguns segundos. Pesquisas recentes indicam que o sistema deverá responder de imediato a qualquer acção do utilizador e portanto dar conhecimento da acção do utilizador. Mais, depois de o sistema tomar conhecimento da acção do utilizador, não deverão passar mais do que 3 a 6 segundos até que o sistema <<actualize>> o utilizador do estado da acção que lhe foi pedida. Por exemplo, se é pedido ao computador que calcule o volume ou a área de um objecto, uma mensagem do género <<a calcular>> não deverá estar "pendurada" no ecrã durante 10 segundos. Uma melhor interface deverá permitir ao utilizador saber o modo como a tarefa está a ser processada.

O tempo de resposta também deverá ser consistente, particularmente quando é necessária a entrada de dados através do teclado. Por exemplo, alguns sistemas armazenam os dados entrados via teclado e depois, com intervalos aleatórios, escrevem os caracteres no ecrã. O utilizador, desconhecendo o processo que está a decorrer, muitas vezes fica confuso à medida que os caracteres vão aparecendo no ecrã com intervalos de tempo que parecem ser aleatórios**[Plastock et al, 1991]**.

Consistência

A interface com o utilizador deverá ser, em todos os aspectos, o mais consistente possível. Informação semelhante deverá sempre aparecer na mesma zona do ecrã. Os resultados das acções do utilizador deverão ter também resultados consistentes. Por exemplo, um sistema vulgarizado requer que o utilizador escreva, em função do nível do diálogo, <<E>>, <<1>>, <<7>> ou <<28>> para sair do sistema. Devido a esta inconsistência, o utilizador é obrigado a tomar nota de qual o nível de diálogo que está activo, ler as opções do menu e depois responder de cada vez que deseja sair**[Plastock et al, 1991]**.

Carga De Trabalho Do Operador

Todos os conceitos discutidos até aqui – Compatibilidade, brevidade, flexibilidade, tempo de resposta e consistência – permitirão reduzir a quantidade de trabalho a cargo do operador. Contudo,

o responsável pelo desenvolvimento do sistema deve também recuar e examinar os tipos específicos de tarefas que o utilizador espera levar a cabo. Em geral, as tarefas gráficas tendem a ser bastante complexas. O criador deverá tentar reduzir a carga de trabalho (1) evitando a redundância, (2) limitando a quantidade de informação em cada momento no ecrã, e (3) transferindo, tanto quanto possível, trabalho do utilizador para o computador.

Muitos dos conceitos ergonómicos até agora discutidos têm sido incorporados nas interfaces com o utilizador de sistemas actuais. Por exemplo, sistemas gráficos de grande qualidade usam agora ícones (representações pictóricas de conceitos) e consistentes distribuições no ecrã. Embora o desenvolvimento de uma boa interface consuma muito tempo e seja cara, a poupança a longo prazo justificará normalmente o esforço**[Plastock et al, 1991]**.

4.4.3 ESTRUTURAS DE INFORMAÇÃO

Plastock et al (1991), Afirmam que um sistema gráfico bem concebido requer uma *estrutura de informação muito dinâmica.* Uma estrutura de informação dinâmica, quando comparada com uma estrutura de informação estática, muda com o evoluir dos cálculos. Por exemplo, uma tabela, que é estática, contém um número fixo de determinado tipo enquanto uma estrutura de informação dinâmica tal como a base de dados de um sistema, muda. Antes de conceber a estrutura de informação, o responsável pelo desenvolvimento do sistema deve examinar cuidadosamente o comportamento típico de um utilizador no desempenho do tipo de tarefas que se espera sejam levadas a cabo. Por exemplo, a estrutura de informação que será usada para fazer esboços será consideravelmente diferente da que será usada para projecto assistido por computador (CAD) ou gráficos de gestão. É importante lembrar que, ao contrário da estrutura de informação que não necessita de seguir o modelo de pensamento do utilizador, a interface necessita. Para conseguir esta exigência, o responsável pelo desenvolvimento tem disponíveis dois métodos básicos para estruturar a informação:

- Toda a informação gráfica deve estar armazenada em memória de visualização, ou
- A informação gráfica deve estar armazenada na forma de um ficheiro de comandos e dados (também chamado *ficheiro de visualização*).

Estruturas De Informação Em Memória De Visualização

Quando toda a informação gráfica estiver armazenada em memória de visualização, o sistema será mais fácil de desenvolver e reagirá mais rapidamente porque não terá de armazenar nem converter comandos e dados para alterar a memória de visualização, Contudo, a edição será limitada por determinadas condições. Por exemplo, suponhamos que o ecrã mostrado na Fig. 4-10 é armazenado apenas em memória de ecrã. Para mover, fazer uma rotação ou executar qualquer outra operação com um dos quadrados, o utilizador terá que apagar selectivamente linhas e depois redesenhar o objecto. Pelo contrário, uma estrutura de informação dinâmica, representando a imagem na forma de um ficheiro de comandos e dados, poderá permitir ao observador identificar o quadrado a ser editado e depois executar a operação de edição. As diferenças entre as tarefas que seriam requeridas ao utilizador estão demonstradas nas Figs. 4-11 (a) à 4-11(c). De notar que em suma, o utilizador tem de executar duas tarefas qualquer que seja a estrutura de informação em que o sistema se baseia. Contudo, a tarefa de apagar é de longe mais demorada e difícil do que a tarefa de identificar**[Plastock et al, 1991]**.

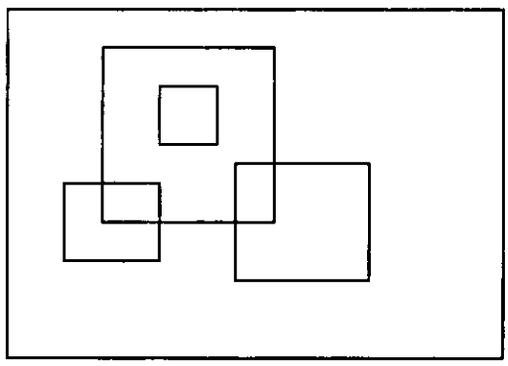


Fig. 4-10 Armazenamento Em Memória de Ecrã.

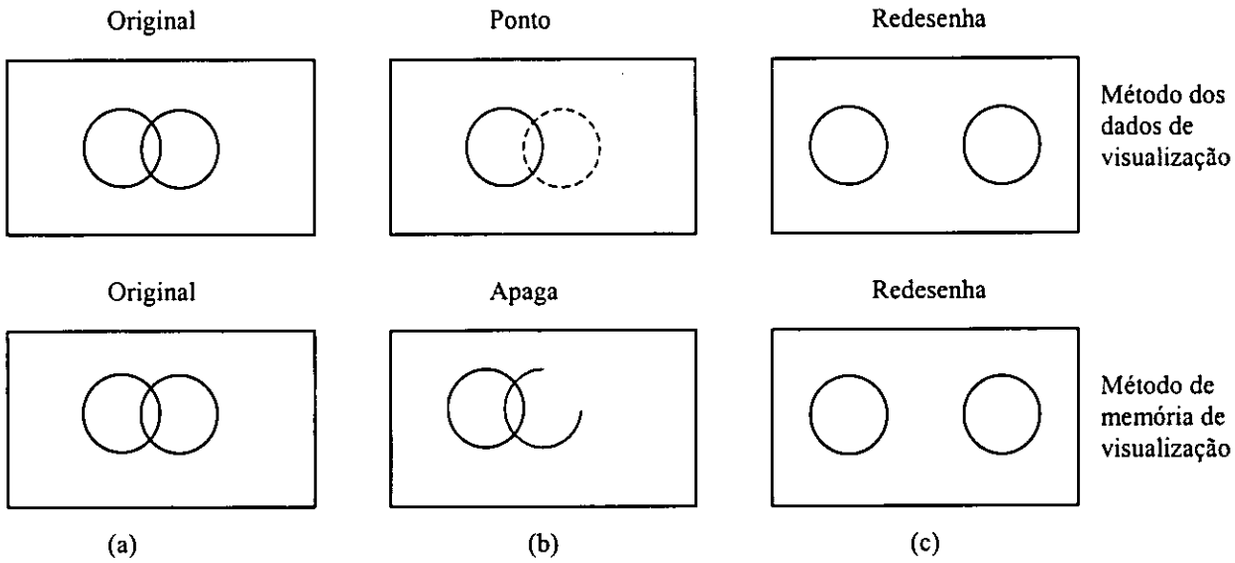


Fig. 4-11 As Diferenças Entre Os Métodos Dos Dados De Visualização E Método De Memória De Visualização.

Estruturas De Informação Em Ficheiro De Visualização

Por razões sugeridas quando falámos da ergonomia, a estrutura de informação dinâmica de comandos e dados será normalmente mais compatível com as necessidades do utilizador.

O ficheiro de comandos e dados (ficheiro de visualização) e a respectiva visualização estão representados nas Figs 4-12(a) a 4-12(c). A visualização de todo o conjunto é dado o nome de <<vista>>. O utilizador desenvolveu a estrutura dinâmica <<casa>> e colocou-a em três locais distintos. Cada casa é composta pelas primitivas <<Triângulo>> e <<Quadrado>>. Para dar ao utilizador um elevado grau de flexibilidade, como será de esperar, a edição deverá ser permitida em diferentes níveis:

- O utilizador deverá poder editar a definição básica de casa, causando assim uma mudança global em cada instância da Casa.
- O utilizador deverá poder editar cada Casa individualmente.
- O utilizador deverá poder formar uma estrutura hierárquica chamada, neste caso, <<vista>>.

A estrutura de informação inicial pode ser representada tal como se mostra na Fig. 4-13(a). De notar que cada ocorrência de Casa é ligada dinamicamente à definição de <<Casa>>. Quando o Utilizador faz uma modificação global, a estrutura será alterada tal como é mostrado na Fig. 4-13(b). Na Fig. 4-13(c) uma instância de Casa sofre uma edição local. De notar que tem de ser criada uma nova

referência, <<Casa 1>>. Casa 1 é agora a entrada de uma nova estrutura que está apenas vagamente ligada à estrutura original Casa, na medida em que estão ambas visualizadas na mesma Vista.

No exemplo da Fig. 4-13(d) toda a imagem é declarada como uma estrutura única , hierárquica, tal como é indicado pela caixa que a envolve, permitindo portanto que as três casas sejam tratadas como um só objecto.

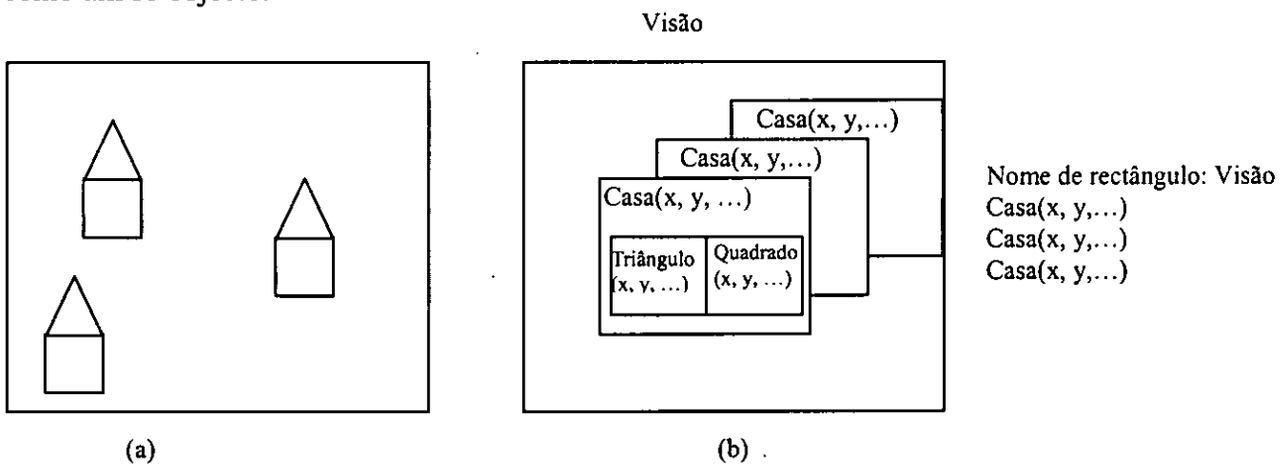
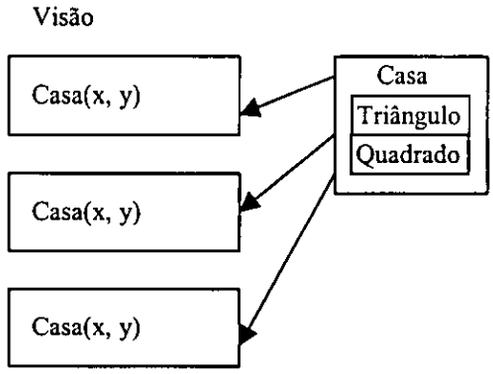
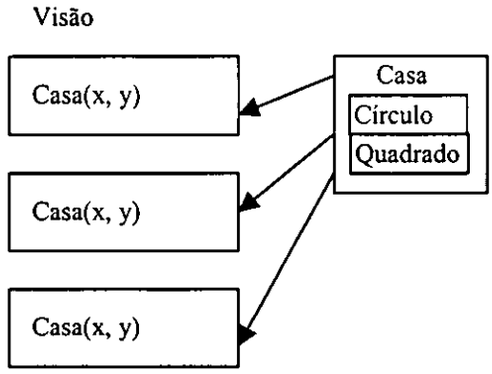


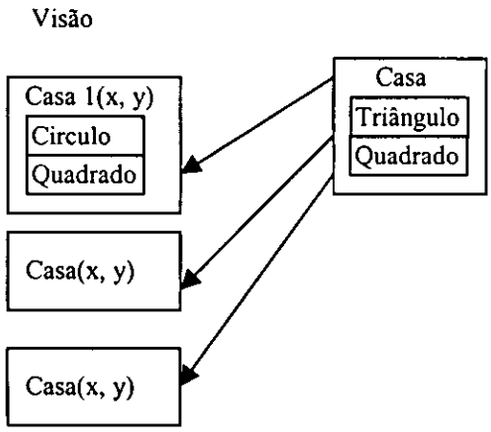
Fig. 4-12 Ficheiro De Comandos E Dados.



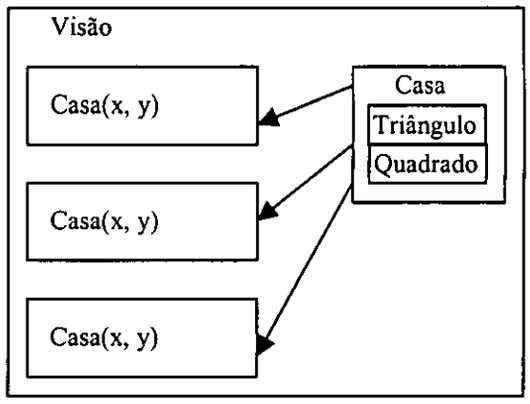
(a) Original.



(b) Após edição global.



(c) Após edição local.



(d) Após declaração de <<visão>> de uma estrutura hierárquica.

Fig. 4-13 Estrutura De Informação.

4.4.4 CAMPO DE APLICAÇÃO

A computação gráfica é normalmente usada para a criação artística, elaboração de gráficos, desenho e projecto ou uma combinação dos três.

Criação Artística

Na verdade, todos os tipos de aplicação envolvem algum grau de criação artística. Nesta parte do trabalho, contudo limitaremos a nossa discussão à tipografia e ao desenvolvimento das apresentações gráficas.

Muitas vezes o utilizador desejará fazer melhorar uma apresentação com imagens, obviamente pretendendo conferir a um diapositivo, transparência ou cópia em papel de um gráfico um aspecto mais profissional. *Viewgraphs* são diapositivos de grandes dimensões, próprios para apresentações feitas com um retroprojector. Uma *sequência de diapositivos* feitos em computador tanto pode ser convertida de modo a ser mostrada em diapositivos correntes (criados através da exposição de uma película de filme à saída de um sistema gráfico) ou armazenada em disco e mostrada num ecrã de um computador.

Independentemente do tipo de saída usado, o utilizador pretenderá desenvolver apresentações sem ajuda de um artista experimentado. Normalmente, o utilizador necessitará das seguintes funções para atingir esse fim:

- *Objectos básicos.* O utilizador desejará, normalmente, ter acesso aos objectos básicos mostrados na Fig. 4-14 (circunferência, elipse, quadrado, rectângulo, linha, linha a traço interrompido e triângulo).
- *Objectos específicos.* Cada área específica e cada instituição que trabalha neste campo terá alguns objectos específicos. A Fig. 4-15, por exemplo, mostra alguns dos diferentes símbolos específicos que serão requeridos por engenheiros electrotécnicos, mecânicos, industriais, químicos e civis.
- *Objectos definidos pelo utilizador.* Tal como foi dito na secção anterior, o utilizador desenvolverá frequentemente estruturas. Estas estruturas representam novos símbolos que o utilizador usará repetidamente numa dada instalação; um exemplo disto é o logotipo de uma empresa.

- *Edição básica.* O utilizador que cria uma representação gráfica esperará conseguir apagar linhas, encher ou imprimir (com padrões), mover, apagar, fazer rotações, variações focais, esticar e reflectir objectos no ecrã.
- *Desenho.* Deverá existir a possibilidade de desenhar à mão livre. Além disso, o utilizador deveria poder mudar a espessura e a inclinação da caneta.

Apetrechando com estas ferramentas, o utilizador deverá ser capaz de criar desenhos razoavelmente profissionais em computador.

O utilizador desejará frequentemente incluir etiquetas, títulos e outros tipos de gráficos que requerem texto. Muitos sistemas oferecem capacidades tipográficas, que permitirão ao utilizador construir virtualmente qualquer fonte, com controlo total sobre o tamanho dos caracteres, inclinação, ascendentes, descendentes e outras características tipográficas. Ainda que tais sistemas sejam altamente especializados, todos os sistemas gráficos deverão dar acesso ao utilizador a diversos tipos de fontes e permitir algum grau de modificação das fontes**[Plastock et al, 1991]**.

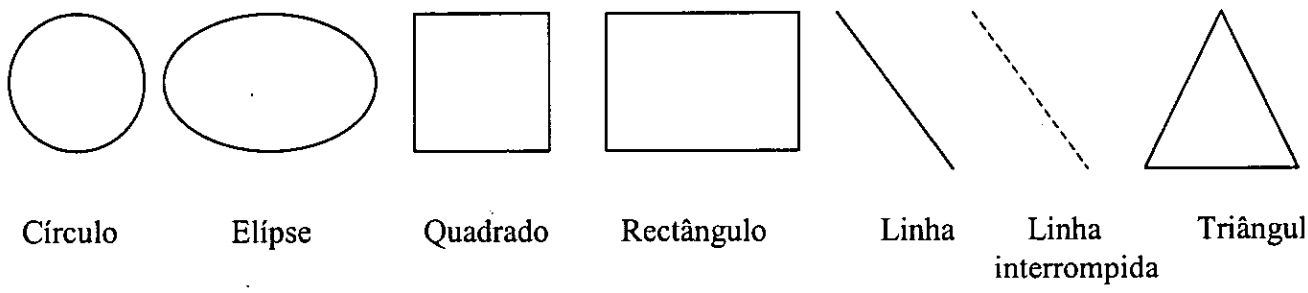


Fig. 4-14 Objectos Básicos.

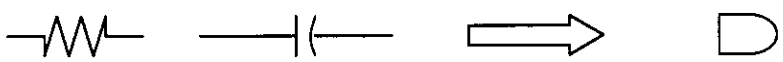


Fig. 4-15 Símbolos Electrotécnicos.

Gráficos

O utilizador pretende, muitas vezes, desenvolver diferentes tipos de gráficos. Os tipos básicos de gráficos são os seguintes: Gráfico circular, gráfico circular explodido, gráfico de barras e gráfico de linhas**[Plastock et al, 1991]**.

Desenho E Projecto

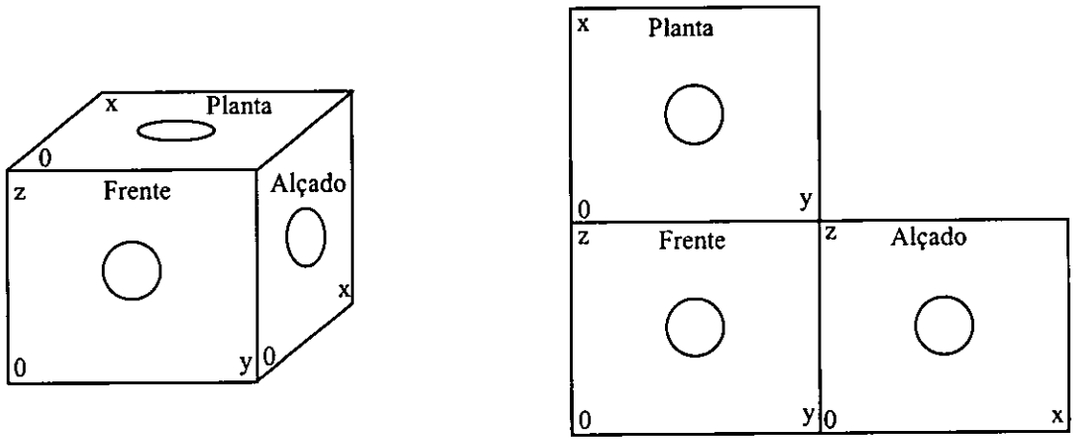
Os engenheiros necessitam em geral, para além das possibilidades de fazer gráficos e da criação artística, de fazer desenho específico, descrição de peças e análise funcional. Os engenheiros querem um sistema que possa mostrar uma peça através de várias projecções. Embora o sistema de computação possa facilmente transformar objectos tridimensionais, é difícil conceber uma interface com a qual o utilizador se sinta confortável.

A interface com a qual o desenhador se sentirá em geral mais à vontade será a que se baseia na técnica do *paralelipípedo transparente* ou de projecção normalmente usada em projecto (ver Fig. 4-16(a) e 4-16(b)). Com esta técnica o utilizador trabalhará num sistema de três coordenadas - três cursores. À medida que o utilizador movimenta o dispositivo que faz a entrada de dados, o movimento é simultaneamente projectado em três planos.

Os modernos sistemas gráficos para a engenharia permitem ao engenheiro ou ao desenhador definir mais do que uma simples peça. Quando uma peça é definida, é também permitido que o engenheiro conceba, efectivamente com o sistema, a peça. Tais sistemas, chamados sistemas para *projecto assistido por computador* (CAD), combinam a listagem de matérias (uma listagem de componentes) e a simulação com as ferramentas de computação gráfica. Isto é conseguido relacionando o desenho da peça com os seus atributos físicos e as instruções de fabrico. Quando todas as instruções de fabrico estão relacionadas com o objecto, a peça definida pode ser directamente fabricada com máquinas de *controlo numérico computadorizado* (CNC). Tais sistemas são chamados *sistemas de projecto e fabrico assistido por computador* (CAD/CAM) ou sistemas de fabrico integrado por computador (CIM).

Uma das possibilidades mais vulgarmente oferecidas pelos sistemas de CAD é a possibilidade de fazer *análise com elementos finitos*. Isto permite ao projectista simular o comportamento mecânico de uma estrutura através da sua modelação como um número finito de componentes mais simples (elementos) cujos estados podem ser mais facilmente calculados do que seria possível simulando

directamente o comportamento da estrutura como um todo (Ver Fig. 4-17). Estas simulações permitem ao responsável pelo desenvolvimento visualizar graficamente os resultados de uma forma facilmente compreensível**[Plastock et al, 1991]**.



(a) Objecto. (b) Projecções do Objecto.

Fig. 4-16 Objecto e Suas Projecções.

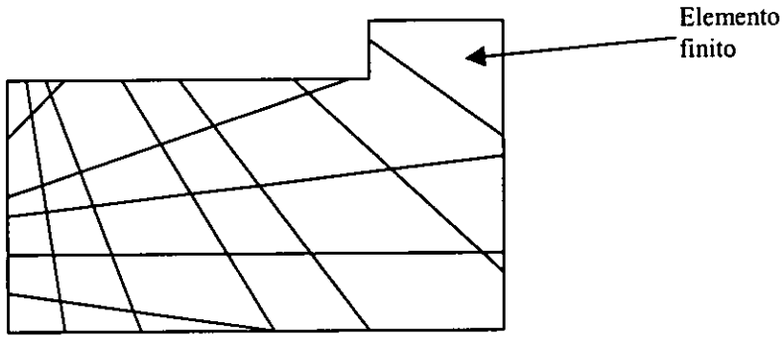


Fig. 4-17 Análise Com Elementos Finitos.

5 SUPERFÍCIE Z=F(X,Y)

5.1 INTRODUÇÃO

Nesta parte do trabalho, serão discutidas algumas definições, formas de geração, classificações, ordem, nomenclaturas, etc., relativas às superfícies. A superfície Z=F(X,Y) é a forma explícita de qualquer superfície dada na forma algébrica, pelo que todos os conceitos que são usados para classificar as superfícies no geral também são válidos para esta superfície. De referir que nem sempre é possível representar a equação de uma superfície na forma Z=F(X,Y).

Também será descrito o algoritmo de desenho de superfícies dadas na forma Z=F(X,Y), que é usado pela aplicação desenvolvida no âmbito deste trabalho.

5.2 DEFINIÇÃO E GERAÇÃO

Podemos definir superfície como o lugar geométrico das posições de uma linha indeformável ou não, chamada geratriz, que se move no espaço seguindo uma lei. Superfície pode ser considerada uma película infinitamente delgada, que separa um corpo do resto do espaço. Há diferença entre superfície e corpo. Um corpo é sempre um volume finito e determinado, enquanto que superfície é a envoltura imaterial que rodeia o corpo.

As superfícies podem apresentar infinitas e variadas formas que dependerão das leis que regem o movimento e mudança de forma da geratriz as quais se orientam, geometricamente por outras linhas ou superfícies chamadas directrizes ou superfícies directoras.

As superfícies podem ser limitadas ou ilimitadas; entre as primeiras estão as fechadas, as quais englobam um volume finito e dão lugar a um corpo**[Ricca, 1992]**.

Dada a enorme variedade de superfícies, somente são estudadas as que têm uma lei de geração conhecida e perfeitamente determinada.

5.3 NOMENCLATURA

Segundo **Ricca (1992)**, podem ser definidos os seguintes termos para o tratamento das superfícies:

- ***Geratriz*** – Linha móvel que gera a superfície; pode ser uma recta ou uma curva; a curva pode ser indeformável ou deformável. Também se designa por geratriz uma posição específica da linha móvel.
- ***Directriz*** – Linha indeformável e fixa no espaço com a qual a geratriz mantém contacto no movimento de geração.
- ***Plano Director*** – plano relativamente ao qual a geratriz (Curva plana ou recta), no movimento de geração, se mantém paralela.
- ***Superfície regrada*** – diz-se duma superfície que pode ser gerada por uma recta em movimento.
- ***Superfície Planificável*** – diz-se duma superfície que se pode desenrolar sobre um plano sem sofrer rupturas ou enrugamentos.
- ***Superfície Empenada*** – diz-se duma superfície que não é possível planificar.
- ***Superfície Curva*** – diz-se duma superfície que não pode ser gerada por uma recta.
- ***Superfície de Revolução*** – diz-se duma superfície cuja geratriz se move em rotação em torno dum eixo; a geratriz pode ser uma recta ou uma curva indeformável.
- ***Superfície Helicoidal*** – diz-se duma superfície cuja geratriz se move em movimento helicoidal; os pontos da geratriz, que pode ser uma recta ou curva indeformável, descrevem hélices cilíndricas com o mesmo eixo e o mesmo passo.
- ***Corda*** – Segmento de recta definido por dois pontos da superfície não pertencentes à mesma geratriz rectilínea.

5.4 CLASSIFICAÇÃO, ORDEM

Não existe um critério geométrico que permita uma classificação detalhada das superfícies. Atendendo à forma da geratriz, as superfícies podem ou não ser geradas por uma recta; as geradas por uma recta podem ou não ser planificáveis. Dada a importância que têm, nomeadamente em aplicações técnicas, a geração por uma recta e a planificação, são considerados, neste trabalho, os grupos de superfícies (que se excluem mutuamente):

REGRADAS PLANIFICÁVEIS;

REGRADAS EMPENADAS E

CURVAS OU NÃO REGRADAS.

A engenharia recorre com grande frequência às superfícies de revolução e às superfícies helicoidais; umas e outras distribuem-se por aqueles três grandes grupos.
Designa-se por ordem duma superfície o número máximo de pontos em que pode ser atravessada por uma recta; por exemplo o plano é uma superfície de primeira ordem**[Ricca, 1992]**.

5.4.1 SUPERFÍCIES REGRADAS PLANIFICÁVEIS

5.4.1.1 DEFINIÇÃO

Só as superfícies regradas podem ser planificáveis. Demonstra-se em geometria analítica ser condição necessária e suficiente para uma superfície ser planificável que do movimento da sua geratriz recta resultem sempre posições infinitamente próximas que sejam complanares. Esta condição exclui do grupo das <<planificáveis >> as superfícies geradas por uma recta que se apoia, no seu movimento, em três directrizes**[Ricca, 1992]**.

5.4.1.2 PROPRIEDADES

Podem se desenvolver sobre um plano, sem que se deforme nenhum de seus elementos. Daí o nome de Planificáveis. Como plano de desenvolvimento pode-se tomar qualquer plano tangente à superfície. Duas geratrizes infinitamente próximas se cruzam, determinando um plano, que é para a superfície o mesmo que o elemento rectilíneo é para a curva, isto é, constitui um elemento superficial plano, sendo ao mesmo tempo, o plano tangente à superfície.

O plano tangente à superfície em um ponto, é também tangente à ela ao longo de toda a geratriz, que passa pelo dito ponto e ao qual contém**[Ricca, 1992]**.

5.4.2 SUPERFÍCIES REGRADAS EMPENADAS

O movimento da recta que gera uma superfície regrada fica determinado por três directrizes d_1, d_2 e d_3, curvas ou rectas, como as da Fig. 5-1: as duas superfícies cónicas, tendo como vértice comum o ponto P da directriz d_1, e como directrizes d_2 e d_3 , têm em comum a recta g, a qual intersecta as três directrizes e gera a superfície quando P se desloca sobre d_1.

Uma das directrizes poderá ser a recta do infinito de um plano; então g move-se paralelamente ao plano; as superfícies assim geradas admitem um plano director.

As superfícies geradas por uma recta obrigada a três directrizes ou a duas directrizes e um plano director não são planificáveis, visto ser a planificação uma condição restritiva do movimento da geratriz**[Ricca, 1992]**.

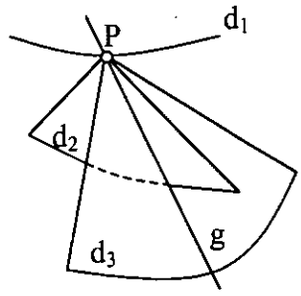


Fig. 5-1 Exemplo de Uma Superfície Regrada Empenada.

5.4.2.1 DEFINIÇÃO

Demonstra-se em geometria analítica que não é planificável a superfície gerada pela deslocação duma recta quando duas posições infinitamente próximas da geratriz, resultantes da deslocação, não são complanares**[Ricca, 1992]**.

5.4.2.2 PROPRIEDADES

- Não podem ser desenvolvidas sobre um plano.
- Duas geratrizes infinitamente próximas se cruzam.
- plano tangente à superfície em um ponto contém a geratriz que passa pelo dito ponto, mas não é tangente à superfície em outros pontos da geratriz citada.

5.4.3 SUPERFÍCIES CURVAS OU NÃO REGRADAS

As quádricas são superfícies algébricas[2] mais simples, visto que as coordenadas dos seus pontos satisfazem a uma equação f(x,y,z) = 0 do segundo grau. Prova-se em Geometria Analítica a existência de sete tipos de quádricas:

$$\text{Re}\,gradas\begin{cases} planificáveis\begin{cases} Cone \\ Cilindro \end{cases} \\ empenadas\begin{cases} Hiperpolóide\ regrado \\ Parabolóide\ regrado \end{cases} \end{cases}$$

$$Curvas(não\ regradas)\begin{cases} Elipsóide \\ Parabolóide\ elíptico \\ Hiperbolóide\ elíptico \end{cases}$$

Existe semelhança geométrica entre o cone e o cilindro: este é um cone com o vértice no infinito. Também existe semelhança geométrica entre o hiperbolóide e o parabolóide regrados: este último é uma hiperbolóide com uma directriz no infinito**[Ricca, 1992]**.

[2] Uma superfície diz-se *algébrica* quando as coordenadas dos seus pontos, referidos a sistema de eixos Oxyz, satisfazem a uma equação f(x,y,z) = 0; as coordenadas dos pontos de planos satisfazem a equações do primeiro grau e as dos pontos das superfícies algébricas mais simples, a equações do segundo grau; tais superfícies designam-se *quádricas*; o grau da equação que representa uma superfície algébrica é também a sua ordem. Distribuem-se pelos três grupos já referidos (Regradas planificáveis, regradas empenadas e Curvas ou não regradas), os vários tipos de quádricas, as quais podem ou não ser <<de revolução>> mas nunca superfícies helicoidais.

5.5 DESCRIÇÃO DO ALGORITMO DO HORIZONTE FLUTUANTE

Segundo **Herigstad et al (1999)**, este algoritmo é usado para desenhar gráficos de funções contínuas de 2 variáveis: Z=F(X, Y). (Fig. 5-2).

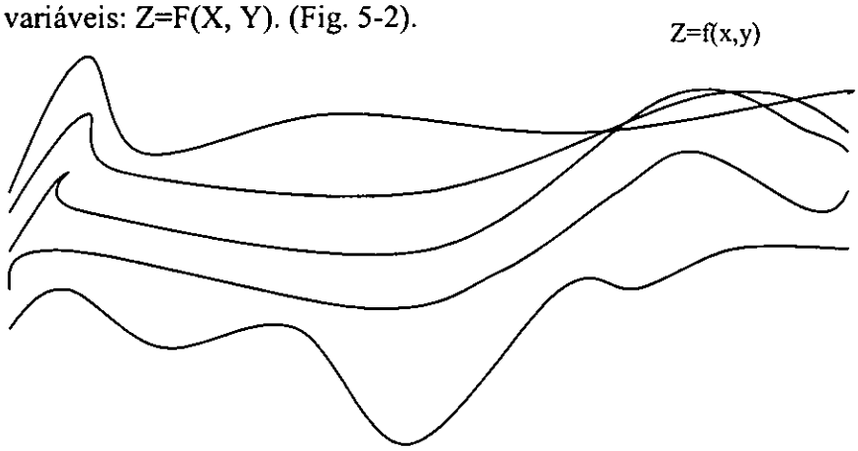
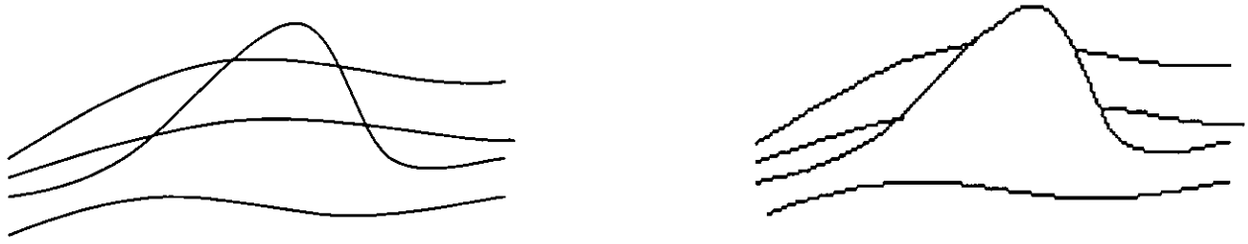


Fig. 5-2 Superfície Z=F(X,Y)

Herigstad et al (1999), afirmam que a filosofia principal deste algoritmo consiste na conversão de um problema tridimensional em bidimensional.

1. Intersectar a superfície com vários planos paralelos em valores constantes de x, y ou z.
   - Ordenar os planos Y = Constante, pelas suas distâncias em relação à janela de visualização.
   - Gerar a curva começando pelo plano mais próximo da janela de visualização. Para cada coordenada x, o valor de z é calculado.

2. Se para um dado valor de x, o valor de z na curva do plano corrente é maior que o valor de z para qualquer curva considerada anteriormente nesse valor de x, então, a curva é visível (ver Figs. 5-3(a) e (b)).



(a) (b)

Fig. 5-3 Visibilidade.

3. Para enquadrar as curvas que são mais profundas que a primeira, um valor para guardar o horizonte mínimo é mantido, valor esse que é mudado à medida que o algoritmo progride (ver Fig. 5-4).

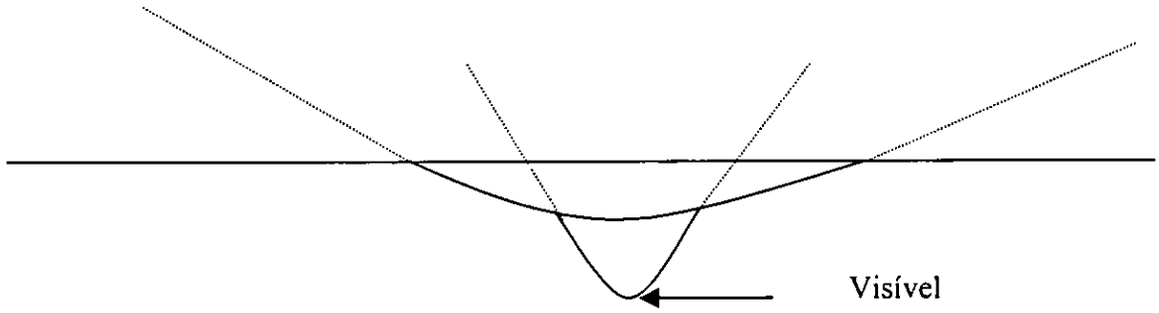


Fig. 5-4 Visibilidade Nas Curvas Mais Profundas

Assim o algoritmo passa para :

- Se para um dado valor de x, o valor de z na curva do plano corrente é maior que o máximo valor de z ou, menor que o mínimo valor de z para qualquer curva considerada anteriormente nesse valor de x, então, a curva é visível.

4. Existe um outro problema ainda por resolver neste algoritmo: visibilidade parcial de uma linha (ver Fig. 5-5).

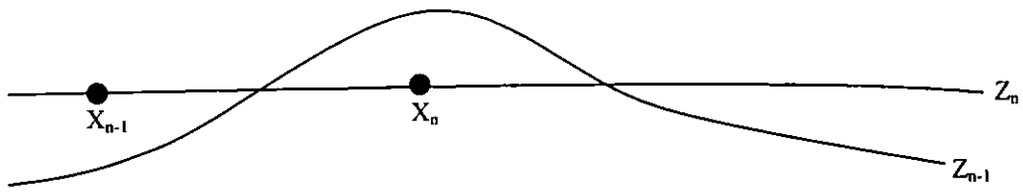


Fig. 5-5 Determinação Da Visibilidade Parcial.

A resolução deste problema consiste em calcular a intersecção entre as curvas Z_n e Z_{n-1} (Fig. 5-6):

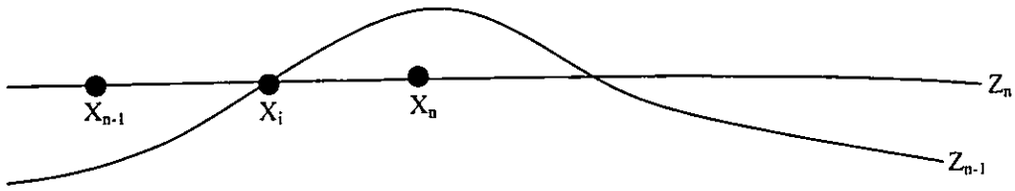


Fig. 5-6 Cálculo Da Intersecção.

5. Assim, com todas as considerações, o algoritmo do horizonte flutuante enuncia-se:

- **Se para um dado valor de x, o valor de z na curva do plano corrente é maior que o máximo valor de z ou, menor que o mínimo valor de z para qualquer curva considerada anteriormente nesse valor de x, então, a curva é visível. Caso contrário a curva é invisível;**
- **Se a linha que vem de um valor anterior de X (X_n) para o valor corrente de X (X_{n+k}), torna-se visível ou invisível, então, calcular a intersecção X_i ;**
- **Desenhar o segmento que vai de X_n até X_{n+k} se o segmento é totalmente visível, de X_n até X_i se ele torna-se invisível, ou de X_i até X_{n+k} se o segmento torna-se visível e**
- **Actualizar os horizontes flutuantes (Máximo e mínimo) desse valor de X. [Herigstad et al, 1999]**

5.5.1 ANÁLISE DA EFICIÊNCIA DO ALGORITMO

O algoritmo do horizonte flutuante opera basicamente sobre dois ciclos de repetição, podendo ser resumido da seguinte maneira:

For Linha := 0 **to** NumeroDeLinhas **DO**
Begin
    <Acções> $O(1)$
    For Ponto:= 0 **to** NumeroDePontos **Do**
    Begin
        <Acções> $O(1)$
    End;
End;

Então, sua eficiência pode ser calculada da seguinte maneira:

$$T(N) = \sum_{i:=0}^{NumLInhas} (O(1) \sum_{j:=0}^{Numpontos} O(1)) = \sum_{i:=0}^{NumLInhas} \sum_{j:=0}^{Numpontos} O(1) = \sum_{i:=0}^{NumLInhas} (O(1)(Numpontos - 0 + 1) =$$

$$= \sum_{i:=0}^{NumLInhas} (O(Numpontos + 1) = O(\sum_{i:=0}^{NumLInhas} (Numpontos + 1)) =$$

$$= O(Numpontos + 1)(Numlinhas + 1) =$$

$$= O(NumPontos*NumLinhas + NumPontos + NumLinhas + 1)$$

Se NumPontos = NumLinhas = N, então,

$T(N) = O(N*N + N + N + 1) = O(N^2 + 2*N + 1) = O(N^2)$.

6 FUNCIONAMENTO E OUTPUTS DA APLICAÇÃO QUE DESENHA A SUPERFÍCIE Z=F(X, Y)

Ao se executar o programa será apresentada a capa do trabalho, a qual indica o autor, o supervisor e o tema do trabalho, para abandonar-se a capa bastará o simples pressionamento de uma tecla qualquer, o que permitirá a visualização do menu principal do trabalho (Fig.6-1).

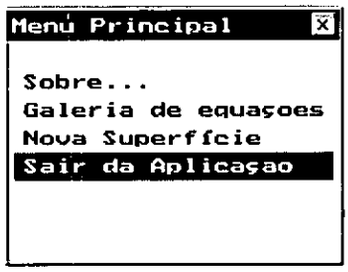


Fig.6-1 Menu Principal.

Para mover-se o cursor pelas opções podem-se usar as teclas de setas ou o mouse e para seleccionar-se uma opção faz-se um clique com o botão direito do mouse ou pressiona-se a tecla ENTER.

6.1 MENU "SOBRE..."

Este menu visualiza uma janela (ver Fig. 6-2) que explica os propósitos da aplicação.

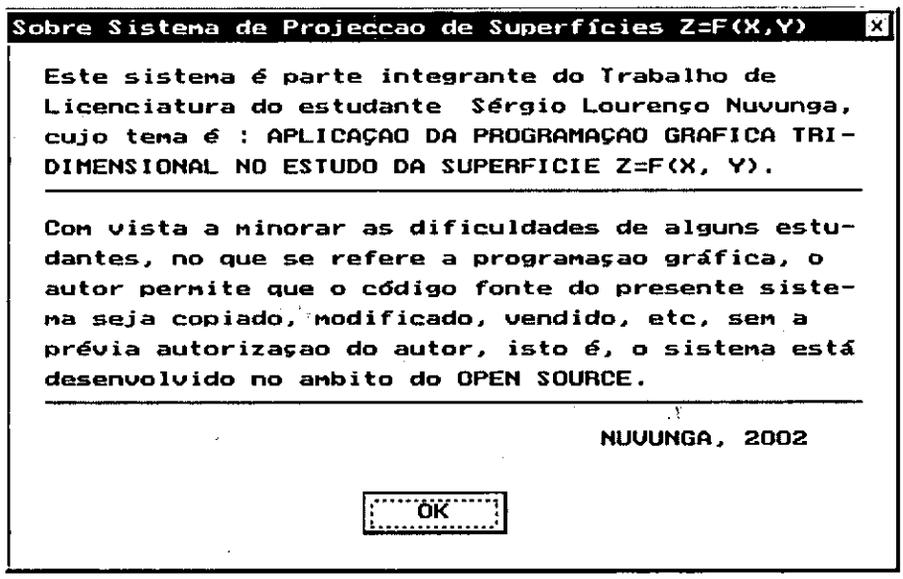


Fig. 6-2 Janela "Sobre...".

6.2 MENU "GALERIA DE EQUAÇÕES"

Este menu permite que equações gravadas num ficheiro sejam visualizadas. Ao seleccionar-se esta opção do menu é apresentada uma janela (ver Fig. 6-3) contendo todas equações do ficheiro, para visualizar-se o gráfico de uma determinada equação, podem ser usadas as teclas de setas para escolher a equação ou através de um click sobre a equação pretendida, e a seguir a tecla ENTER ou um click sobre o botão "visualizar". Para cancelar-se a exibição da janela deve ser feito um click sobre o botão "cancelar" ou o botão "x", o mesmo pode ser feito a partir do teclado através da tecla TAB, até seleccionar-se o botão "cancelar" e em seguida pressionar-se a tecla ENTER.

Se a opção "visualizar" for seleccionada, será aberta uma janela contendo uma gama de opções, onde pode-se escolher o tipo de projecção a visualizar (Ver Fig. 6-5).

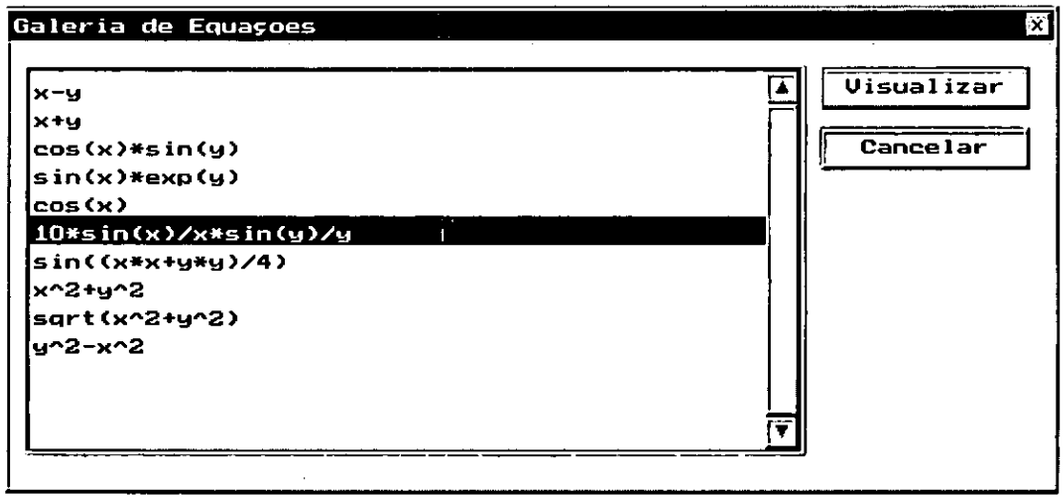


Fig. 6-3 Galeria De Equações.

6.3 MENU "NOVA SUPERFÍCIE"

Ao seleccionar-se esta opção do menu é apresentada uma janela (Fig. 6-4), que permite a introdução da equação, seu domínio e parâmetros para o esboço do gráfico(número de linhas e número de pontos), para além disso, a partir desta janela é possível a visualização do gráfico da equação e sua gravação para o ficheiro de equações.

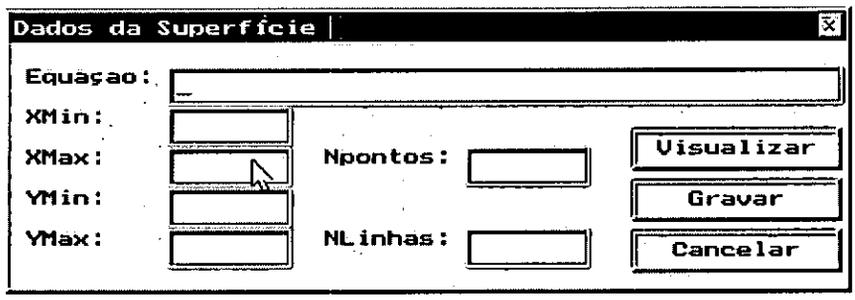


Fig. 6-4 Janela para a introdução de dados, visualização e gravação.

6.4 MENU "SAIR DO PROGRAMA"

Esta opção do menu permite ao utilizador abandonar a aplicação. Esta tarefa também pode ser executada através do pressionamento da tecla ESC ou através do botão "X".

6.5 JANELA "PROJECÇÃO"

Esta janela (Fig. 6-5) é exibida quando é pressionado o botão visualizar, a partir dos menus "Galeria de Equações" e "Nova Superfície". Se a opção escolhida for a primeira (Perspectiva...), antes de ser visualizado o gráfico da superfície, é apresentada uma janela (Fig. 6-6) para a introdução da distância do observador (Rho) e dos ângulos de visão (Theta e Phi). Se a opção escolhida for a segunda (Paralela qualquer...), antes da visualização do gráfico da superfície, é apresentada a janela da Fig. 6-7, para a introdução dos ângulos da projecção (Theta e Phi). Para as restantes opções (Dimétrica e Isométrica), o gráfico é visualizado de imediato.

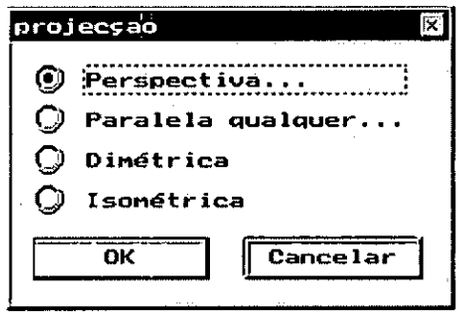


Fig. 6-5 Janela "Projecção".

Fig. 6-6 Janela "Rho e ângulos de Projecção".

Fig. 6-7 Janela "Ângulos de Projecção".

6.6 EXEMPLOS DE GRÁFICOS GERADOS PELA APLICAÇÃO

A seguir apresentam-se alguns exemplos de gráficos de superfícies geradas pela aplicação. É muito evidente que com o auxílio desta aplicação o trabalho de desenho de superfícies da forma Z=F(X, Y) fica bastante simplificado.

Exemplo 1

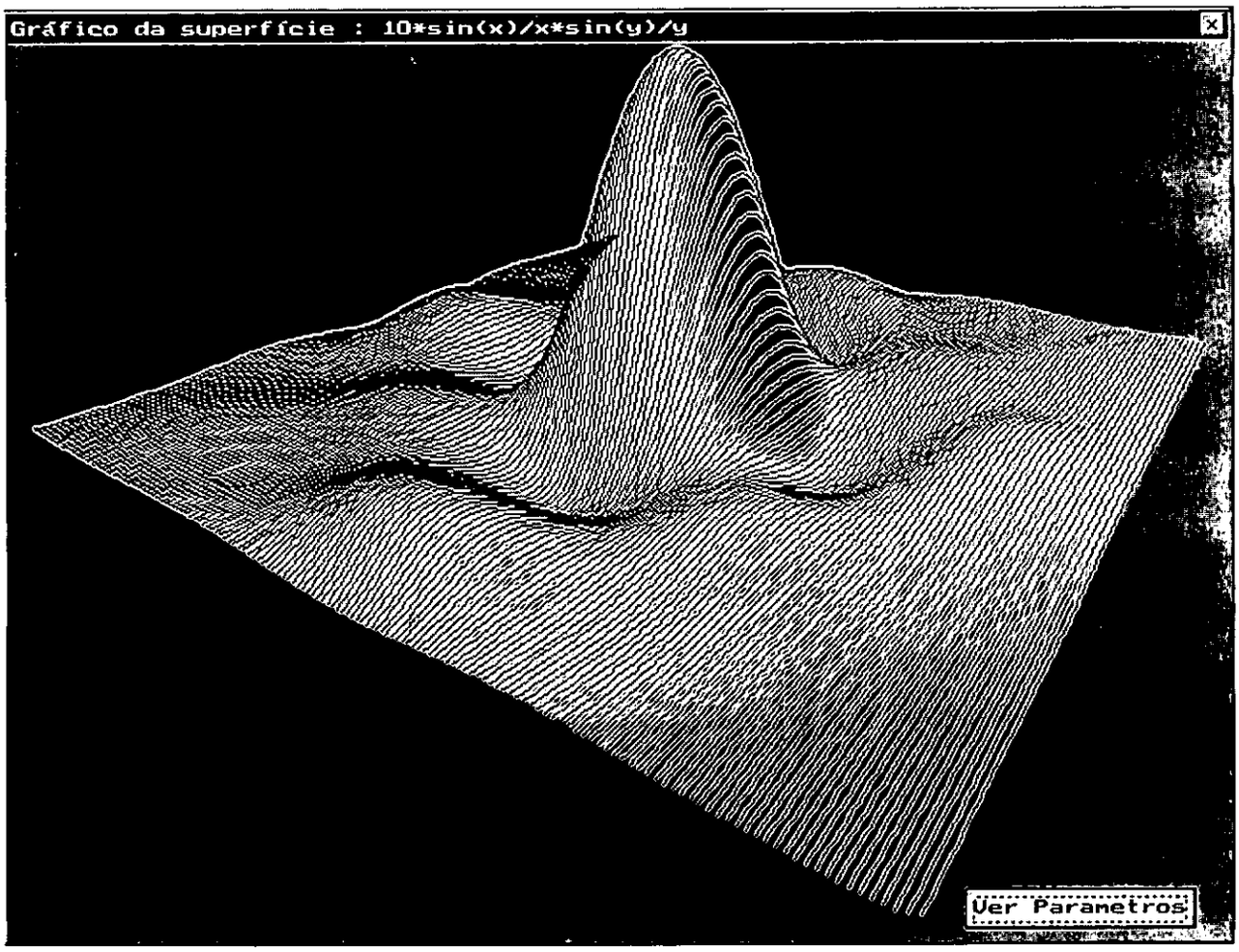


Fig. 6-8 Gráfico da Superfície Z=10*Sin(X)/X*Sin(Y)/Y

Ao "clicar-se" o botão "ver parâmetros" é visualizada uma janela (Fig. 6-9) contendo todos os parâmetros da superfície.

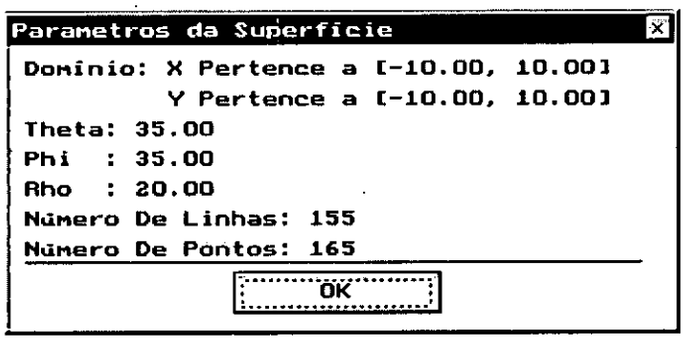


Fig. 6-9 Parâmetros da superfície Z=10*Sin(X)/X*Sin(Y)/Y

Exemplo 2

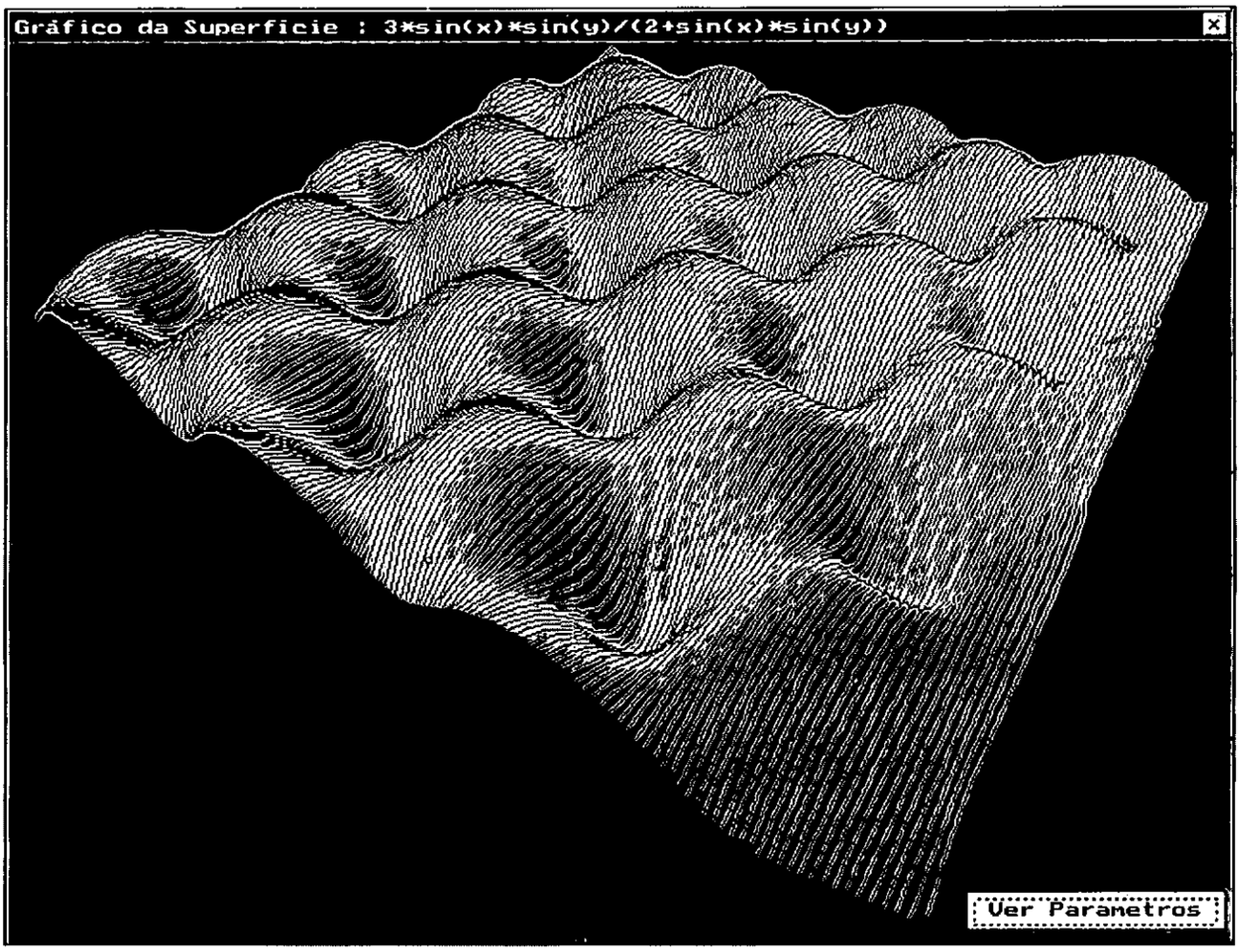


Fig. 6-10 Gráfico da Superfície Z = 3 * Sin(X) * Sin(Y) / (2 + Sin(X)*Sin(Y))

Exemplo 3

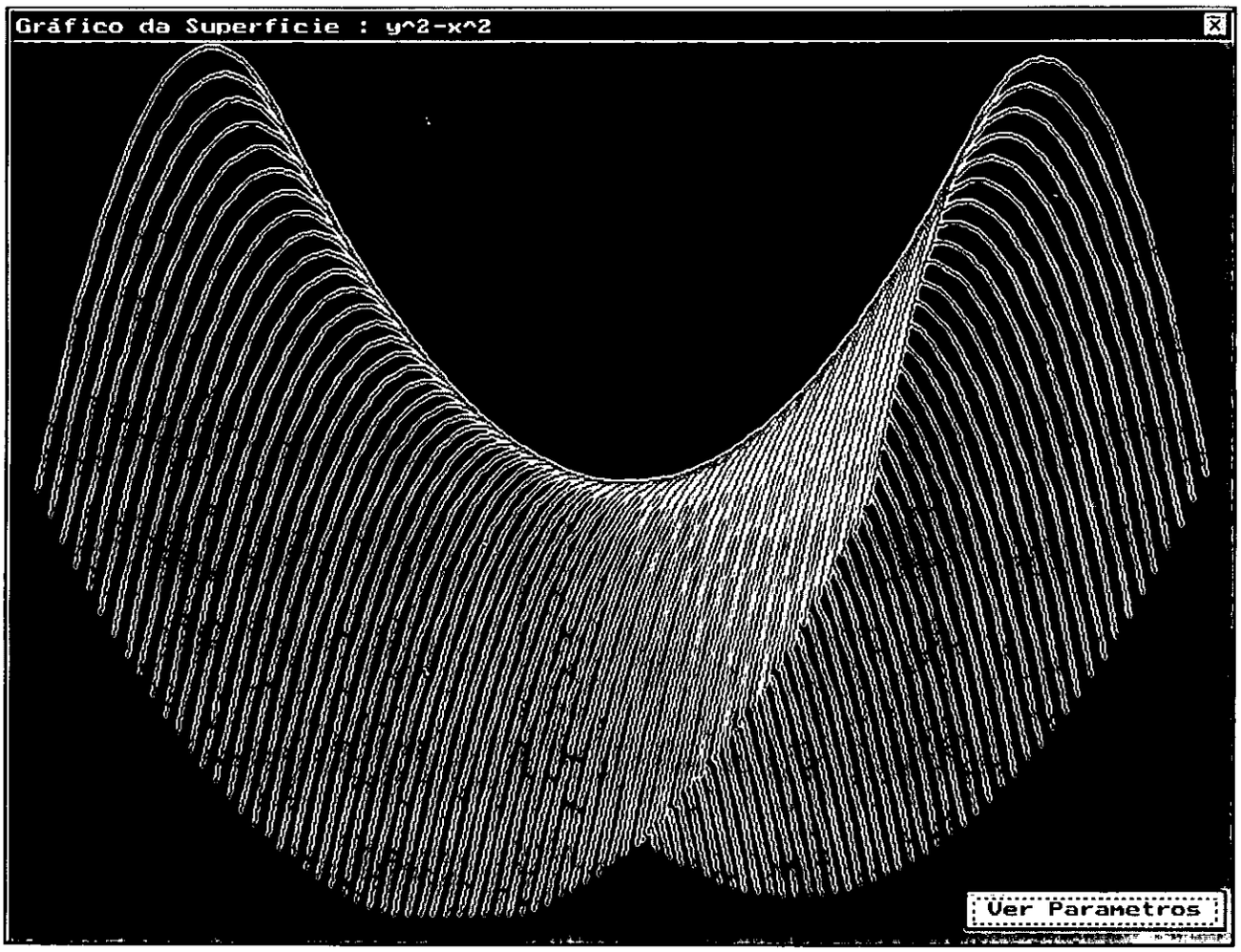


Fig. 6-11 Gráfico da Superfície $Z = Y^2 - X^2$

7 DISCUSSÃO

Neste trabalho estão descritos muitos conceitos de programação gráfica tridimensional, suas áreas de aplicação, assim como as características principais das superfícies no geral. Na maior parte, os objectivos definidos estão cumpridos, exceptuando-se a parte que lida com os conceitos sobre superfícies, o autor constatou que existe uma escassez de material bibliográfico que descreve detalhadamente as superfícies.

Sobre a aplicação desenvolvida, esta possui os seus pontos fortes e fracos. Dos pontos fortes destacam-se:

- A aplicação permite a geração de gráficos de superfícies da forma Z=F(X,Y), num interface gráfico fácil de utilizar e bastante simples. Esse interface permite a introdução de expressões analíticas de superfícies, seus parâmetros (Domínio, Nº de pontos e Nº de linhas), a gravação destes dados, bem como a visualização do respectivo gráfico. É também, um ponto forte da aplicação, o facto de a expressão ser validada momento da sua leitura.

Como pontos fracos, destacam-se:

- A aplicação utiliza para o desenho do gráfico, um algoritmo com uma complexidade de ordem quadrática ($T(N)=O(N^2)$), tornando a exibição do gráfico bastante demorada. Dependendo da precisão que se deseja, essa demora pode levar até cerca de cerca de 20 segundos. Também considera-se ponto fraco da aplicação, o facto de o domínio não ser validado; pois, se o utilizador introduzir um domínio em que a função não é definida, a aplicação aborta abrutamente a sua execução.

8 CONCLUSÕES E RECOMENDAÇÕES

8.1 CONCLUSÕES

Ao iniciar este trabalho o autor se propunha a:

- Criar um sistema informático capaz de desenhar as superfícies da forma Z = F(X, Y) e
- Criar uma biblioteca de funções e procedimentos da unidade gráfica do Turbo Pascal.

Terminado que está o trabalho, conclui que os objectivos do trabalho foram cumpridos na íntegra.

No presente trabalho foram estudados os seguintes conceitos:

- Programação gráfica;
- Superfícies e
- Algoritmo para o desenho de superfícies Z = F(X, Y).

Desse estudo conclui-se que a *programação gráfica* exige sólidos fundamentos de geometria analítica bi- e tridimensional, combinados com o conhecimento de estruturas de dados e algoritmos. Conclui-se também que não existe um critério geométrico que permita uma classificação detalhada das superfícies**[Ricca, 1992]**. Contudo, elas podem ser classificadas em: *Regradas Planificáveis, Regradas Empenadas* ou *curvas (não regradas).* Quanto ao algoritmo usado para o desenho da superfície Z = F(X, Y), o *algoritmo do horizonte flutuante*, possui uma eficiência na ordem de $O(n^2)$.

Quanto ao sistema para o desenho de superfícies Z = F(X, Y), conclui-se que ele desenha qualquer tipo de superfície Z = F(X, Y), ele oferece um bom interface gráfico com o utilizador, ele corre no ambiente MS-DOS.

8.2 RECOMENDAÇÕES

Para o melhoramento deste trabalho, recomenda-se o seguinte:

- Que seja melhorado o tempo de resposta do procedimento que calcula o valor de uma função dada sob a forma de um string;
- Que seja desenvolvido um módulo para a validação do domínio da equação da superfície;
- Que seja desenvolvido um módulo para a impressão da superfície Z = F(X, Y) no papel;

- Que seja feita uma conversão deste programa para uma linguagem de 32 bits por ex., o Delphi, o Visual Basic, etc;

9 BIBLIOGRAFIA

9.1 BIBLIOGRAFIA REFERENCIADA

- **[Plastock et al, 1991]** Plastock, R. A., G. Kalley (1991). Computação Gráfica. 414 pp. Portugal. McGraw-Hill.
- **[Ricca, 1992]** Ricca, G. (1992). Geometria Descritiva – Método De Monge. 353 pp. Lisboa. Fundação Calouste Gulbenkian.
- **[Macome, 1995]** Macome, E.(1995). Introdução à metodologia de Investigação, Maputo, UEM.
- **[O'brien, 1992]** O'brien, S.(1992). Turbo Pascal 6 - Completo e Total, São Paulo, Makron, McGraw-Hill.
- **[Weiskamp et al, 1989]** Weiskamp, K., L. Heiny, N. Shammas(1989). Power Graphics Using Turbo Pascal. 365 pp. USA. John Wiley & Sons, Inc.
- **[Herigstad et al, 1999]** Herigstad, G., G. Lindahl (1999). Hidden Line & Surfaces. USA, http://www.vrac.iastate.edu/~carolina/519/notes/6.hidden.pdf

9.2 BIBLIOGRAFIA NÃO REFERENCIADA

- Ammeraal L. (1995). Programming Principles In Computer Graphics. 2nd Edition. 233 pp. England. John Wiley & Sons, Inc.
- Http://www.mat.uel.br/geometrica/geome/portoguese/portal/portalgd.htm(2000) Geometria Descritiva. Consultada no dia 10/10/2002.
- Http://cmaf.lmc.fc.ul.pt/em_accao/superficies_regradas/(2000) Superfícies Regradas. Consultada no dia 17/10/2002.
- Nguyen Ngoc Quynh(2003). Unidade Calcul.tpu. DMI, UEM, Maputo.

10 ANEXOS

10.1 PROGRAMA PRINCIPAL

```
Program DesenhaSuperficies;
{Elaborado Por S. L. NUVUNGA, 2002

Este programa desenha superfícies da forma z = f(x, y), o usuário tem a possibilidade de introduzir
a equação que deseja visualizar apartir do teclado ou seleccionar uma já existente apartir do menu
galeria}
uses
    Estetica, sobre, Galeria, dadosup1, graph, gpoppac9, mousenuv;

const
    Nopcoes = 4;
    Opcoes : Array[1..nopcoes] of string[40] = ('Sobre... ',
                                                'Galeria de equações ',
                                                'Nova Superfície',
                                               { 'Imprimir Superfície',}
                                                'Sair da Aplicação');
    larguradasopcoes = 160;{comprimento dos titulos do menu principal}

Type
    JanelaMouse = Record
      x1, y1, x2, y2 : Integer;
End;
{esta estrutura de dados guarda as coordenadas dos rectangulos das opcoes do menu, usada
para verificar se o mouse está  ou não sobre uma determinada opção}
  MatrizJanelas = array[1..Nopcoes] of JanelaMouse;
{este array guarda todos os rectangulos que constituem as opções do menu}

Const
    JanelasMouse : MatrizJanelas = ((x1:5; y1:31; x2:5 + larguradasopcoes; y2:44),
                                    (x1:5; y1:45; x2:5 + larguradasopcoes; y2:58),
```

```
                                    (x1:5; y1:59; x2:5 + larguradasopcoes; y2:72),
                                    (x1:5; y1:73; x2:5 + larguradasopcoes; y2:86){,
                                    (x1:5; y1:87; x2:5 + larguradasopcoes; y2:100)});
        {esta constante guarda todas as coordenadas dos rectangulos que constituem as
         opcoes do menu principal}

Var
   j : Integer;

Var
   gdriver : Integer;
   gmode :Integer;  {parametros da unidade graph}
   fim1, mudouDeOpcao  : Boolean;
   Code1 : Integer;{variavel que guarda o código da tecla lida}
   I : Integer;
   Opcao : Byte; {opcao escolhida}
   Ptr : Pointer; {apontador que , usado para gravar a  rea de uma determinada opcao}
   Size : Integer;

Function MouseInOption(JanelasMouse : MatrizJanelas; x, y : Integer; Var J : Integer) : Boolean;
{verifica se o mouse está  numa determinada opção ou não}
Var
    fim1 : Boolean; st : string;

Begin
      J := 1; fim1 := False;
      While (j <= Nopcoes) and not fim1 do
      If  ((x >= JanelasMouse[j].x1 + 234) and (x <= JanelasMouse[j].x2 + 234)) and
          (y >= JanelasMouse[j].y1 + 175) and (y <= JanelasMouse[j].y2 + 175) then
          fim1 := true else j := j + 1;
          MouseInOption := fim1;
End;

procedure windowerror;
```

```
begin
    closegraph;writeln('failed to open window');halt(1);
end;

Begin
    gdriver := detect;
    initgraph(gdriver, gmode, 'c:\bp\bgi');
    If not MouseInstalled then outtext('nao existe mouse') else
    Mouseinit;
    MouseShow;
    capa;
    mousehide;
    setfillstyle(9, 8);
    bar(0, 0, getmaxx, getmaxy);
    mouseshow;
    fim1 := false;
    mousehide;
    if not gpopup('Menu Principal', 234,175, getmaxx-234,getmaxy-175, solidln, 15, solidfill, 7)
then
    windowerror;
    {desenhar janela do menu}
    SetColor(0);
    size := imagesize(5, 31, larguradasopcoes + 5, 44);
    getmem(Ptr, size); {arranjar espaço na memória}
    For i := 1 to nopcoes do
    Outtextxy(10, i * 14 + 20, Opcoes[i]);{escrever todas opções}
    opcao := 1;
    Getimage(5, 31, 5 + larguradasopcoes, 44, Ptr^);{captar a area ocupada pela 1ª opção
                                                    e guardar no apontador PTR}
    SetColor(15);
    mudouDeOpcao := true;
    Repeat
        setfillstyle(solidfill, blue);
        If mudouDeOpcao then
```

```
Begin
        Bar(JanelasMouse[opcao].x1, JanelasMouse[opcao].y1,
        JanelasMouse[opcao].x2, JanelasMouse[opcao].y2);
        Outtextxy(10, opcao * 14 + 20, Opcoes[opcao]);
        mudouDeOpcao := false;
end;
mouseshow;
Code1 := waitforinput(leftbutton);
{Ficar a espera do botão esquerdo do mouse ou teclado}
mousehide;
If (code1 = 72)  then  {Seta para cima}
Begin
        Putimage(JanelasMouse[opcao].x1, JanelasMouse[opcao].y1, Ptr^, CopyPut);
        If Opcao > 1 Then Opcao := Opcao - 1 Else Opcao := NOpcoes;
        mudouDeOpcao := true;
        Getimage(JanelasMouse[opcao].x1, JanelasMouse[opcao].y1,
        JanelasMouse[opcao].x2, JanelasMouse[opcao].y2,  Ptr^);
End else if code1 = 80 then {Seta para baixo}
Begin
        Putimage(JanelasMouse[opcao].x1, JanelasMouse[opcao].y1, Ptr^, CopyPut);
        If Opcao < Nopcoes Then Opcao := Opcao + 1 Else Opcao := 1;
        mudouDeOpcao := true;
        Getimage(JanelasMouse[opcao].x1, JanelasMouse[opcao].y1,
        JanelasMouse[opcao].x2, JanelasMouse[opcao].y2,  Ptr^);
End Else if code1 = 13 then {A tecla enter foi pressionada}
Begin
        Case Opcao of
        1 : about; {chamar o procedimento about da unidade SOBRE}
        2 : galeria_de_equacoes; {Chamar o procedimento galeria_de_equacoes
                        da unidade GALERIA}
        3 : begin
                mousehide;
                BarraDeTitulo('Menu Principal', 0,0, getmaxx-234-234,getmaxy-175-
                175, solidfill, 8);
```

```
                EntradaDeDadosPrincipais;
                {chamar o procedimento EntradaDeDadosPrincipais da unidade
                dadosup1}
                BarraDeTitulo('Menu Principal', 0,0, getmaxx-234-234,getmaxy-175-
                175, solidfill, 1);
                mouseshow;
            end;
            Nopcoes : fim1 := True;
        End;
    End Else if code1 = -1 then {o mouse foi clickado}
    Begin
        If (MouseInBox(getmaxx - 234 - 16, 175+3 , getmaxx - 234 - 4,175+ 15,
        MouseXPos, MouseYPos)) or (MouseInBox(JanelasMouse[Nopcoes].x1 + 234,
        JanelasMouse[Nopcoes].y1 + 175, JanelasMouse[Nopcoes].x2 + 234,
        JanelasMouse[Nopcoes].y2 + 175, MouseXPos, MouseYPos))
        {O mouse devolve xpos e ypos como valores da janela (0,0,getmaxx,getmaxy),
        por isso e necessário ajustar os valores das coordenadas da janela corrente +234
        para abcissa e + 175 para a ordenada onde 234 e 175 correspondem a x1 e y1 da
        janela corrente}
        Then fim1 := true else
        If (MouseInBox(JanelasMouse[3].x1 + 234, JanelasMouse[3].y1 + 175,
                JanelasMouse[3].x2 + 234, JanelasMouse[3].y2 + 175,
                MouseXPos, MouseYPos))
        Then EntradaDeDadosPrincipais else
        If (MouseInBox(JanelasMouse[2].x1 + 234, JanelasMouse[2].y1 + 175,
                JanelasMouse[2].x2 + 234, JanelasMouse[2].y2 + 175,
                MouseXPos, MouseYPos))
        Then galeria_de_equacoes else
        If (MouseInBox(JanelasMouse[1].x1 + 234, JanelasMouse[1].y1 + 175,
                JanelasMouse[1].x2 + 234, JanelasMouse[1].y2 + 175,
                MouseXPos, MouseYPos))
        Then about;
End Else If code1 = 300 then {o mouse foi movimentado}
    Begin
```

```
                If (MouseInOption(JanelasMouse, mousexpos, mouseypos, j)) and (j <> opcao)
                then
                Begin
                        Putimage(JanelasMouse[opcao].x1, JanelasMouse[opcao].y1, Ptr^,
                        CopyPut);
                        opcao := j;
                        mudouDeOpcao := true;
                        getimage(JanelasMouse[j].x1, JanelasMouse[j].y1,
                        JanelasMouse[j].x2, JanelasMouse[j].y2,  Ptr^);
                End;
        End;
        If (code1 = 27) then fim1 := true;
    Until fim1;
    gunpop(234,175, getmaxx-234,getmaxy-175);
End.
```

10.2 UNIDADE GRAPH3D

```
Unit Graph3d;

Interface

Uses
    printer, dos, crt, graph;
const
    Escala = 1.35;

Type
    TipoDeProjeccao = (perspectiva, paralela);
Var
   rho, theta, phi, DE : real;
   aux1, aux2, aux3, aux4 : real;
   aux5, aux6, aux7, aux8 : real;
   projeccao : TipoDeProjeccao;
   Xobs, Yobs, Zobs : real;
   Xproj, Yproj : real;
   XEcran, YEcran : integer;
   GraphDriver, GraphMode : integer;
   maxx, maxy : integer;
   NumeroDeCores : byte;

  Procedure EcranGrafico;{Inicializa o modo Gráfico}
  Procedure EcranTextual;{Volta ao modo Texto}
  Procedure ApagarEcran;{Limpa o ecrã Gráfico}
  Procedure MudarCor(Cor : byte);{Atribui a cor indicada por cor}
  Procedure Visualizacao(c1, c2, c3, c4 : integer);{Define a área de visualização da informação}
  Procedure InicialisarProjeccao;{Faz a inicialização do tipo e dos parâmetros da projecção}
  Procedure Projectar(x, y, z : real);{Projecta o ponto x:y:z para um determinado tipo de
projecção}
  Procedure TracarPara(x, y, z : real);{Traça uma linha para o ponto x:y:z}
```

```
Procedure MoverPara(x, y, z : real);{Move o cursor para o ponto x:y:z}
Procedure Eixos;{Desenha os eixos de coordenadas}
Procedure LegendaDosEixos;{Coloca uma legenda nos eixos de coordenadas}
Procedure Esperar;{Fica a espera do pressionamento de uma tecla}
Procedure Beep;{Emite um beep}

Implementation

Procedure EcranGrafico;
Var
   ErrorCode : integer;

Begin
      GraphDriver:=detect;
      InitGraph(GraphDriver, GraphMode, 'C:\bp\BGI');
      ErrorCode := GraphResult;
      if errorcode <> grOK then
      Begin
            writeln('Erro : ',GraphErrorMsg(errorcode));
            halt(1);
      End;
      maxx := getmaxx;
      maxy := getmaxy;
      NumeroDeCores := getmaxcolor;
End;

Procedure EcranTextual;
Begin
     Closegraph
End;

Procedure ApagarEcran;
Begin
      cleardevice
```

```
End;

Procedure MudarCor(Cor : byte);
Begin
    setcolor(Cor);
End;

Procedure Visualizacao(c1, c2, c3, c4 : integer);
Begin
    Setviewport(c1, maxy - c4, c2, maxy - c3, clipon);
End;

Procedure InicialisarProjeccao;
Var
  th, ph : real;

Begin
    th := pi * theta / 180.0;
    ph := pi * phi / 180.0;
    aux1 := sin(th);
    aux2 := sin(ph);
    aux3 := cos(th);
    aux4 := cos(ph);
    aux5 := aux3 * aux2;
    aux6 := aux1 * aux2;
    aux7 := aux3 * aux4;
    aux8 := aux1 * aux4;
End;

Procedure Projectar(x, y, z : real);

Begin
    Xobs := -X * aux1 + Y * aux3;
    Yobs := - X * aux5 - Y * aux6 + Z * aux4;
```

```
    if projeccao = perspectiva then
    Begin
        Zobs := -X * aux7 - Y * aux8 - Z * aux2 + rho;
        Xproj := DE * Xobs / Zobs;
        Yproj := DE * Yobs / Zobs;
    End else
    Begin
        Xproj := DE * Xobs;
        Yproj := DE * Yobs;
    End;
End;

Procedure TracarPara(x, y, z : real);

Begin
    Projectar(x, y, z);
    XEcran := round(Xproj * Escala + maxx div 2 );
    YEcran := round(maxy div 2 - Yproj);
    lineto(XEcran, YEcran);
End;

Procedure MoverPara(x, y, z : real);

Begin
    Projectar(x, y, z);
    XEcran := round(Xproj * Escala + maxx div 2);
    YEcran := round(maxy div 2 - Yproj);
    moveto(XEcran, YEcran);
End;

Procedure Eixos;
Begin
    MoverPara(0, 0, 0); TracarPara(200, 0, 0);
    MoverPara(0, 0, 0); TracarPara(0, 200, 0);
```

```
        MoverPara(0, 0, 0); TracarPara(0, 0, 200);
End;

Procedure LegendaDosEixos;
Begin
        Setlinestyle(dottedln, 0, normwidth);
        MoverPara(0, 0, 0); TracarPara(3, 0, 0);
        Moverel(5, 5); Outtext('X');
        MoverPara(0, 0, 0); TracarPara(0, 3, 0);
        Moverel(5, 5); Outtext('Y');
        MoverPara(0, 0, 0); TracarPara(0, 0, 3);
        Moverel(5, 5); Outtext('Z');
        SetlineStyle(solidln, 0, normwidth);
End;

Procedure Esperar;
Begin
        sound(400); delay(200); nosound;
        repeat until keypressed;
End;

Procedure beep;
Begin
        sound(2000); delay(50); nosound;
end;

End.
```

10.3 UNIDADE SUPERFIC – UNIDADE RESPONSÁVEL PELO DESENHO DA SUPERFÍCIE

```
Unit Superfic;
{Unidade reponsável pelo esboço da superfície Z=F(x,y)}
Interface
        Uses crt, graph, graph3d, calcul1, mousenuv;
Const
    limX = {640;}610;{VGA}
    limY = {480;}450;
Type
    table = array[0..limX] of integer;
Var
  Hmax, Hmin, tabaux : table;
  x1, x2, y1, y2 : real;
  incX, incY : real;
  f1, f2, f3, f4 : real;
  c1, c2, c3, c4 : integer;
  echx, echy, ech : real;
  Vue : Char;
  xg, yg, xd, yd : integer;
  NumeroDeLinhas, NumeroDePontos : integer;
  Funcao : string;

Procedure InicializacoesDiversas;{Faz a inicialização dos diversos parâmetros da aplicação}
Procedure CalcularJanela;{Calcula as dimensões da área de visualização}
Procedure CalcularEscalas;{Determina as escalas necessárias para a visualização do gráfico}
Procedure DesenharFuncao;{Faz o desenho do gráfico da superfície}


Implementation
Function signe(x:real):integer;
Begin
    if x > 0 then signe := 1 else
```

```
        if x < 0 then signe := -1 else signe := 0
End;

Procedure Entrada;
Begin
        DE := 1;
        projeccao := paralela;
End;

Procedure InicializacoesDiversas;
{Inicializações diversas}
Var
    aux : real;
    i : integer;
Begin
        incx := (x2 - x1) / NumeroDePontos;
        incy := (y2 - y1) / NumeroDeLinhas;
        c1 := 0; c2 := limx - 1; c3 := 1; c4 := limy;
        f1 := 1E37; f2 := -1E37; f3 := 1E37; f4 := -1E37;
        xg := -1; yg:=-1; xd := -1; yd := -1;
        fillchar(hmax, sizeof(hmax), 0);
        for i := 0 to limx do hmin[i] := limy;
        if (theta < 0) or (theta > 180) then
        begin
       aux := x1; x1 := x2; x2 := aux; incx := -incx;
       aux := y1; y1 := y2; y2 := aux; incy := -incy;
  End;
End;
Procedure CalcularJanela;
Var
    x, y, z : real;
    linha, ponto : integer;

Begin
```

```
        for linha := 0 to NumeroDeLinhas do
        begin
            y := y2 - linha * incy;
            for ponto := 0 to NumeroDePontos do
            begin
                x := x1 + ponto * incx;
                calcul1.Valorf(funcao, x, y, 0, z);
                Projectar(x, y, z);
                if Xproj < f1 then f1 := Xproj;
                if Xproj > f2 then f2 := Xproj;
                if Yproj < f3 then f3 := Yproj;
                if Yproj > f4 then f4 := Yproj;
            end
        end
End;

Procedure CalcularEscalas;
{Calcular Escalas}
Begin
        echx := (c2 - c1) / (f2 - f1);
        echy := (c4 - c3) / (f4-f3);
        if Upcase(vue) = 'R' then if echx < echy then echy := echx else
        echx := echy;
End;

Procedure Horizon(x1, y1, x2, y2 : integer);
{Algoritmo do horizonte flutuante}
Var
    x, y, dx : integer;
    Declive : real;

  Function max(x1, x2 : integer) : integer;
  Begin
     if x1 < x2 then max := x2 else max := x1
```

```
  End;

  Function min(x1, x2 : integer) : integer;
  Begin
     if x1 < x2 then min := x1 else min := x2
  End;

Begin
      dx := signe(x2 - x1);
      if dx = 0 then
      begin
            hmax[x2 + 1] := max(hmax[x2], y2);
            hmin[x2 + 1] := min(hmin[x2], y2);
      end else
      begin
            Declive := (y2 - y1) / (x2 - x1);
            for x := x2 + 1 to x1 do
            begin
                  y := round(Declive * (x - x1) + y1);
                  hmax[x] := max(hmax[x], y);
                  hmin[x] := min(hmin[x], y);
            end;
      end;
end;

Procedure Visibilidade(x, y : integer; Var visi : shortint);
{Procedimento que determina a visibilidade de um ponto}
Begin
      if (y < hmax[x]) and (y > hmin[x]) then visi := 0
      else if y >= hmax[x] then visi := 1 else visi := -1
end;

Procedure Inter(x1, y1, x2, y2 : longint;{Var} tabaux : table; Var Xi, Yi : integer);
{Procedimento que determina a intersecção entre duas linhas}
```

```
Var
    den, xii, yii : real;
Begin
      if x2 - x1 = 0 then
      begin
            xii := x2; yii := tabaux[x2]
      end
      else begin
                  den := y2 - y1 - tabaux[x2] + tabaux[x1];
                  if den <> 0 then
                  begin
                        xii := (x1 * (y2 - tabaux[x2]) + x2 * (tabaux[x1] - y1)) / den;
                        yii := (y2 * tabaux[x1] - y1 * tabaux[x2]) / den;
                  end  else
                  begin
                        xii := x2;
                        yii := y2
                  end
         end;
         xi := round(xii);
         yi := round(yii);
End;

Procedure Contorno(x, y : integer; Var Xlateral, ylateral : integer);
Begin
      if xlateral <> -1 then horizon(xlateral, ylateral, x, y);
      xlateral := x;
      ylateral := y
End;

Procedure DesenharFuncao;
{Procedimento que desenha a função}
Var
    xe, ye : integer;
```

```
    linha, ponto : integer;
    xAnterior, yAnterior : integer;
    xCorrente, yCorrente : integer;
    xi, yi : integer;
    visiCorrente, visiAnterior : shortint;
    x, y, z : real;

Begin
    setcolor(0);
    for linha := 0 to NumeroDeLinhas do
    begin
        mousehide;
        y := y2 - linha * incy;
        x := x1;
        calcul1.Valorf(funcao, x, y, 0, z);
        Projectar(x, y, z);
        XAnterior := round((xproj - f1) * echx) + c1;
        YAnterior := round((yproj - f3) * echy) + c3;
        Contorno(XAnterior, YAnterior, xd, yd);
        moveto(XAnterior, limy - YAnterior);
        Visibilidade(XAnterior, YAnterior, VisiAnterior);
        For ponto := 0 to NumeroDePontos do
        begin
            x := x1 + ponto * incx;
            calcul1.Valorf(funcao, x, y, 0, z);
            Projectar(x, y, z);
            XCorrente := round((Xproj - f1) * echx) + c1;
            YCorrente := round((yproj - f3) * echy) + c3;
            visibilidade(XCorrente, YCorrente, visiCorrente);
            if (hmax[XCorrente] = 0) or (hmin[XCorrente] = limy) then
            visiCorrente := VisiAnterior;
            if visiCorrente = VisiAnterior then
            begin
                if (visiCorrente = 1) or (visiCorrente = -1) then
```

```
            begin
                line(XAnterior, limy - YAnterior, XCorrente, limy - YCorrente);
                horizon(XAnterior, YAnterior, XCorrente, YCorrente);
            end
        end else
        begin
            if visiCorrente = 0 then
            begin
                if VisiAnterior = 1 then
                inter(XAnterior, YAnterior, XCorrente, YCorrente, hmax, xi, yi)
                else inter(XAnterior,YAnterior,XCorrente,YCorrente,hmin,xi,yi);
                line(XAnterior,limy-YAnterior,xi,limy-yi);
                horizon(XAnterior, YAnterior, xi, yi)
            end  else
            begin
                if visiCorrente = 1 then
                begin
                if VisiAnterior = 0 then
                begin
                    inter(XAnterior, YAnterior, XCorrente, YCorrente, hmax, xi,
                    yi);
                    line(xi, limy - yi, XCorrente, limy - YCorrente);
                    horizon(xi, yi, XCorrente, YCorrente)
                end  else
                begin
                    inter(XAnterior, YAnterior, XCorrente, YCorrente, hmin, xi,
                    yi);
                    line(XAnterior, limy - YAnterior, xi, limy - yi);
                    horizon(XAnterior, YAnterior, xi, yi);
                    inter(XAnterior, YAnterior, XCorrente, YCorrente, hmax, xi,
                    yi);
                    line(xi, limy - yi, XCorrente, limy - YCorrente);
                    horizon(xi, yi, XCorrente, YCorrente);
                end
```

```
                    end else
                    begin
                        if VisiAnterior = 0 then
                        begin
                            inter(XAnterior, YAnterior, XCorrente, YCorrente, hmin, xi,
                            yi);
                            line(xi, limy - yi, XCorrente, limy - YCorrente);
                            horizon(xi, yi, XCorrente, YCorrente);
                        end else
                        begin
                            inter(XAnterior, YAnterior, XCorrente, YCorrente, hmax, xi,
                            yi);
                            line(XAnterior, limy - YAnterior, xi, limy - yi);
                            horizon(XAnterior, YAnterior, xi, yi);
                            inter(XAnterior, YAnterior, XCorrente, YCorrente, hmin, xi,
                            yi);
                            line(xi, limy - yi, XCorrente, limy - YCorrente);
                            horizon(xi, yi, XCorrente, YCorrente);
                        end
                    end
                end
            end;
            VisiAnterior := visiCorrente;
            XAnterior := XCorrente;
            YAnterior := YCorrente;
        end;
        Contorno(XCorrente, YCorrente, xg, yg);
        mouseshow;
    end ;
End;

End.
```

10.4 DESCRIÇÃO DOS INTERFACES DAS UNIDADES GRÁFICAS NÃO LISTADAS

10.4.1 UNIDADE ANGULOS

- ***IntroduzirAngulos***

Sintaxe:

Procedure IntroduzirAngulos;

Descrição:

Desenha um formulário que permitirá a introdução dos angulos de uma projecção paralela qualquer.

10.4.2 UNIDADE ANGULOS1

- ***IntroduzirAngulos***

Sintaxe:

Procedure IntroduzirAngulos;

Descrição:

Desenha um formulário que permitirá a introdução dos angulos de uma projecção perspectiva.

10.4.3 UNIDADE GALERIA

- ***Galeria_de_equacoes***

Sintaxe:

Procedure Galeria_de_equacoes;

Descrição:

Desenha uma caixa de listagem que visualiza as equações que estão guardadas no ficheiro de equações e dá a possibilidade de visualizar o gráfico de uma determinada equação.

10.4.4 UNIDADE DADOSSUP1

- ***EntradaDeDadosPrincipais***

Sintaxe:

Procedure EntradaDeDadosPrincipais;

Descrição:

Este procedimento desenha um formulário que permite a introdução dos dados da superfície e seus parâmetros.

10.4.5 UNIDADE GPOPPAC9

- ***ProgressBar***

Sintaxe:

Procedure ProgressBar(titulo : string; Tempo : word);

Descrição:

Este procedimento desenha uma barra que indica o progresso da execução de uma tarefa.

- ***desenharFormulario***

Sintaxe:

Procedure desenharFormulario(titulo : string; left, top, right, bottom, bordertype, bordercolor,

backfill, fillcolor : integer);

Descrição:

Este procedimento desenha um formulário que possui um título definido no string **titulo**.

- ***BarraDeTitulo***

Sintaxe:

Procedure BarraDeTitulo(titulo : string; left, top , right, bottom, backfill, fillColor : integer);

Descrição:

Este procedimento desenha uma barra de título de um formulário.

- ***gpopup2***

Sintaxe:

function gpopup2(titulo : string; left, top, right, bottom, bordertype, bordercolor, backfill, fillcolor : integer) : boolean;

Descrição:

Esta função grava o conteúdo de uma área rectangular.

- ***gpopup***

Sintaxe:

function gpopup(titulo : string; left, top, right, bottom, bordertype, bordercolor, backfill, fillcolor : integer) : boolean;

Descrição:

Esta função grava o conteúdo de uma área rectangular e desenha nela um formulário que possui um título indcado pelo string **titulo**.

- ***gunpop***

Sintaxe:

procedure gunpop(Left, Top, Right, Bottom : Integer);

Descrição:

Este procedimento liberta a memória ocupada por uma janela.

- ***Botao2***

Sintaxe:

Procedure Botao2(titulo : string; left, top , backfill, fillColor : integer; cortexto, Estado : Integer);

Descrição:

Este procedimento desenha um botao e indica o estado : (1 : activo, 2 : activo e está com o foco, 3 - não está com o foco).

- ***RadioButton***

Sintaxe:

Procedure RadioButton(Titulo : string; X, Y : Integer; Estado : Integer);

Descrição:

Este procedimento desenha um radio button, escreve o seu título e indica o estado : (1 : activo, 2 : activo e está com o foco, 3 - não está com o foco).

- ***CaixaDeTexto***

Sintaxe:

Procedure CaixaDeTexto(titulo : string;left, top, comprimento, backfill, fillColor : integer);

Descrição:

Este procedimento desenha uma caixa de texto que possui um título, especificado em **titulo**.

- ***Rectangulo***

Sintaxe:

Procedure Rectangulo(left, top, right, bottom , backfill, fillColor : integer);

Descrição:

Este procedimento desenha um formulário sem título.

10.4.6 UNIDADE MOUSENUV

- ***MouseInstalled***

Sintaxe:

Function MouseInstalled : Boolean;

Descrição:

Esta função verifica se o mouse está instalado.

- ***MouseInit***

Sintaxe:

Procedure MouseInit;

Descrição:

Esta função Inicializa o mouse.

- ***MouseShow***

Sintaxe:

Procedure MouseShow;

Descrição:

Este procedimento visualiza o cursor do mouse.

- ***MouseHide***

Sintaxe:

Procedure MouseHide;

Descrição:

Este procedimento esconde o cursor do mouse.

- ***MouseButton***

Sintaxe:

Function MouseButton : Byte;

Descrição:

Esta função devolve o número do botão do mouse que tiver sido pressionado.

- ***MouseXpos***

Sintaxe:

Function MouseXPos : Word;

Descrição:

Esta função devolve o número da coluna em que o mouse foi pressionado.

- ***MouseYpos***

Sintaxe:

Function MouseYPos : Word;

Descrição:

Esta função devolve o número da linha em que o mouse foi pressionado.

- ***MouseGotoXY***

Sintaxe:

Procedure MouseGotoXY(x, y : Word);

Descrição:

Este procedimento envia o cursor do mouse para a posição **x:y**.

- ***MouseWindow***

Sintaxe:

Procedure MouseWindow(x1, y1, x2, y2 : Word);

Descrição:

Este procedimento define uma janela para a qual o mouse está activo.

- ***MouseInBox***

Sintaxe:

Function MouseInBox(x1, y1, x2, y2, x, y : Integer) : Boolean;

Descrição:

Esta função Verifica se o mouse pressionado sobre a área rectangular (x1, y1, x2, y2).

- ***MouseButtonReleased***

Sintaxe:

function MouseButtonReleased(whichbutton : integer) : boolean;

Descrição:

Esta função verifica se um botão do mouse foi largado.

- ***MouseButtonPressed***

Sintaxe:

function MouseButtonPressed(whichbutton : integer) : boolean;

Descrição:

Esta função verifica se um botão do mouse foi Pressionado.

- ***GetInput***

Sintaxe:

function GetInput(whichbutton : integer) : integer;

Descrição:

Permite a leitura do mouse ou teclado.

- ***WaitForInput***

Sintaxe:

function WaitForInput(whichbutton : integer) : integer;

Descrição:

Esta função fica a espera de uma entrada (teclado ou mouse) e dá resultado TRUE se o botão do Mouse é largado ou o mouse é movimentado e ainda se uma tecla é pressionada.

- ***WaitForInput2***

Sintaxe:

function WaitForInput2(whichbutton : integer; var x1, y1, x2, y2 : integer) : integer;

Descrição:

Esta função fica a espera de uma entrada (teclado ou mouse) e dá resultado TRUE se o botão do mouse é largado ou uma tecla é pressionada(**x1:y1** coordenadas do mouse quando se pressiona um botão e **x2:y2** coordenadas do mouse quando se larga um botão).

10.4.7 UNIDADE GTEXT3

- ***IntToStr***

Sintaxe:

Function IntToStr(Num : LongInt) : String;

Descrição:

Esta função faz a conversão de um número inteiro **Num** para um string.

- ***RealToStr***

Sintaxe:

Function RealToStr(n : Real; Width, Decimals : Integer) : String;

Descrição:

Esta função faz a conversão de um número real **n** para um string.

- ***Gwrite***

Sintaxe:

Procedure Gwrite(S : String);

Descrição:

Este procedimento faz a escrita do string S no ecrã.

- ***Gwritexy***

Sintaxe:

Procedure Gwritexy(x, y : Integer; S : String);

Descrição:

Este Procedimento escreve o string S na posição **x:y**.

- ***Gwritech***

Sintaxe:

Procedure Gwritech(Ch : Char);

Descrição:

Este procedimento faz a escrita do Caracter CH no ecrã.

- ***GReadReal***

Sintaxe:

Function GReadReal(Var Num : Real; Var UltimoCharLido : Integer; maxChars : Integer) : Boolean;

Descrição:

Este procedimento permite a leitura de um número real, onde **MaxChars** representa o número máximo de algarismos do número a ser lido e **UltimoCharLido** representa o último caracter lido.

- ***GReadInteger***

Sintaxe:

Function GReadInteger(Var Num : Integer; Var UltimoCharLido : Integer; maxChars : Integer): Boolean;

Descrição:

Este procedimento permite a leitura de um número Inteiro, onde **MaxChars** representa o número máximo de algarismos do número a ser lido e **UltimoCharLido** representa o último caracter lido.

- ***GReadStr***

Sintaxe:

Function GReadStr(Var S : String100; Var UltimoCharLido : Integer; maxChars : Integer) : Boolean;

Descrição:

Este procedimento permite a leitura de um string, onde **MaxChars** representa o número máximo de caracteres do string a ser lido e **UltimoCharLido** representa o código do último caracter lido.