UNIVERSIDADE FEDERAL DE SANTA CATARINA

CURSO DE POS-GRADUAÇAO EM ENGENHARIA MECANICA

EDITOR GRAFICO PARA PROJETO MECANICO

DISSERTAÇAO SUBMETIDA A UNIVERSIDADE FEDERAL DE SANTA CATARINA

PARA A OBTENÇAO DO GRAU DE MESTRE EM ENGENHARIA MECANICA

MAURICIO KUSTER

FLORIANOPOLIS - SC

MAIO DE 1989

EDITOR GRAFICO PARA PROJETO MECANICO

MAURICIO KUSTER

ESTA DISSERTAÇAO FOI JULGADA ADEQUADA PARA OBTENÇAO DO TITULO DE

MESTRE EM ENGENHARIA

ESPECIALIDADE ENGENHARIA MECANICA, AREA DE CONCENTRAÇAO PROJETO, APROVADA EM SUA FORMA FINAL PELO CURSO DE POS-GRADUAÇAO EM ENGENHARIA MECANICA.

Prof. Edison da Rosa, M.Eng.
Orientador

Prof. Arno Blass, Ph.D.
Coordenador do Curso

BANCA EXAMINADORA :

Prof. Edison da Rosa, M.Eng.

Prof. Clovis Sperb de Barcellos, Ph.D.

Prof. Abelardo Alves de Queiroz, Ph.D.

A minha família, que tanto me apoiou e incentivou nesta longa jornada.

AGRADECIMENTOS

Ao professor Edison da Rosa, pela orientação prestada durante o desenvolvimento deste trabalho.

Ao CNPq, pelo apoio financeiro concedido durante grande parte da realização da dissertação.

A Coordenadoria do Curso, e aos professores do Departamento de Engenharia Mecânica da UFSC que, de alguma forma, contribuíram para o desenvolvimento deste trabalho.

A todos os colegas da pós-graduação que me auxiliaram na elaboração do trabalho e, principalmente, pela amizade presente em todos os momentos.

Em especial, aos amigos e colegas Fernando e Eduardo ("Mineiro"), pela amizade e incentivo que nunca me faltaram.

Aos companheiros do GRANTE, professores, alunos e funcionários, especialmente aos graduandos Altemir, Rogério ("Rato"), Arima e Marcelo, pela importante contribuição na elaboração do programa.

INDICE

RESUMO

O Editor Gráfico para Projeto Mecânico é um sistema desenvolvido para ser conectado a aplicativos na área de projeto mecânico. O objetivo do trabalho é a utilização dos recursos gráficos disponíveis nos microcomputadores tipo IBM-PC compatíveis, para tornar mais eficientes os programas de projeto, acoplando a parte gráfica às rotinas de cálculo e dimensionamento. Torna-se possível, então, o projeto de um componente mecânico desde a entrada de dados, até sua saída em forma de desenho para a fabricação.

Como exemplo foi elaborado um programa para dimensionamento de molas helicoidais de compressão. Este programa calcula a mola de forma a minimizar o parâmetro escolhido para projeto. O resultado numérico é convertido em elementos geométricos básicos, como linhas, arcos, círculos, etc, que são gravados num arquivo, chamado arquivo gráfico. Este arquivo é interpretado pelo Editor para gerar o desenho da mola na tela de vídeo ou nos dispositivos gráficos de saída, como impressoras ou traçadores gráficos.

ABSTRACT

The Graphic Editor for Mechanical Design is a system developed to be connected to other application programs for mechanical design. The purpose of this work is to make use of the IBM-PC compatible microcomputers' graphic power in order to give more efficience for the design programs, coupling the graphics with the calculation and dimensioning routines. It's then possible to design a mechanical component since the data input till it's output, as a drafting for the manufacturing.

As an example, it was created a program for helical compression springs design. This program calculates the spring for minimum values of certain design parameters. The numerical results are arranged in basic geometric elements, like lines, arcs, circles, etc, witch are saved in a file, called graphic file. This file is interpreted by the Editor in order to display the spring drafting on the screen or in an output device as a printer or a plotter.

CAPITULO 1

INTRODUÇAO

1.1. Introdução

Imagens e desenhos sempre foram as formas mais naturais de comunicação entre os seres humanos. Da mesma forma, os engenheiros utilizam-se de informações gráficas, na forma de desenhos, para se comunicar. Um desenho ou gráfico substitui, com vantagens, extensos relatórios numéricos ou tabelas de difícil interpretação. O jargão popular "um desenho vale mais que mil palavras" mostra claramente o poder dos recursos gráficos na comunicação.

A Computação Gráfica, surgida com o advento dos computadores, veio a facilitar a geração e tratamento de imagens. Os recursos de tratamento facilitam a alteração de imagens previamente geradas, que permitem a geração de desenhos de melhor qualidade e em menor tempo que em processos manuais, e mesmo a solução de problemas que não seriam possíveis pelos processos tradicionais. Pode-se definir a Computação Gráfica como o conjunto de algoritmos, técnicas e metodologias para

tratamento e representação gráfica de informações, através da criação, armazenamento e manipulação de figuras, utilizando-se computadores e dispositivos periféricos gráficos [16].

Os efeitos combinados dos aumentos dos custos dos materiais, da energia e outros custos gerais, que afetam adversamente a indústria, bem como a necessidade de uma maior qualidade do produto e redução do tempo de desenvolvimento, em virtude da competitividade do mercado, trouxeram uma ênfase bem maior no projeto. Desta forma a Computação Gráfica tem hoje o seu principal campo de aplicação na engenharia, em sistemas de apoio ao projeto e fabricação, chamados Sistemas de *CAE/CAD/CAM*. Sistemas deste tipo estão sendo cada vez mais empregados para reduzir o tempo de desenvolvimento e melhorar a qualidade dos projetos.

No Brasil ainda há muita dificuldade para a aplicação mais efetiva da Computação Gráfica e mesmo de Sistemas para *CAE/CAD/CAM*. Pode-se citar como um dos principais problemas a falta de pessoal especializado, que deve-se à ausência de cursos de Computação Gráfica no currículo das escolas. Outro ponto importante, talvez o principal obstáculo para o desenvolvimento da Computação Gráfica no país, é a falta de opções em termos de computadores e dispositivos gráficos no mercado nacional. Soma-se ainda o fato da S.E.I. ter feito, até pouco tempo atrás, sérias restrições para a importação deste tipo de equipamento, o que tornava muito elevado o custo de Sistemas de *CAE/CAD/CAM*, tanto para o desenvolvimento e pesquisa, como para a produção

industrial. Hoje tem-se uma abertura maior por parte da S.E.I. no que diz respeito a equipamentos específicos para *CAE/CAD/CAM*, tornando possível que algumas instituições de ensino e pesquisa, bem como algumas indústrias adquiram equipamentos deste tipo.

1.2. Histórico

A Computação Gráfica começou a dar seus primeiros passos na década de 50, com o surgimento dos primeiros dispositivos gráficos, como o terminal gráfico (1951), os traçadores gráficos ou *plotters* (1953) e as canetas luminosas ou *light pens* (1958). As primeiras iniciativas de aplicação da Computação Gráfica na indústria apareceram na década de 60, com o trabalho de pesquisa de algumas indústrias de grande porte. Na maioria destes casos estava presente um dos grandes fornecedores de computadores : IBM, Control Data ou DEC.

O grande passo para o avanço da Computação Gráfica foi dado no *Massachusets Institute of Technology - MIT* , em 1962, com a apresentação da tese de doutoramento de Ivan E. Sutherland, "*Sketchpad : a man-machine graphical communications system*". Sutherland mostrou que gráficos podem ser desenhados e manipulados na tela de um terminal de vídeo tipo CRT, com auxílio de uma caneta luminosa. Estava então viabilizada a criação de programas gráficos interativos, e aí surgiu a Computação Gráfica Interativa. Até então o que se tinha eram programas gráficos passivos, ou seja, os desenhos era gerados de uma só vez, após um certo processamento, sem intervenção do projetista. Para

modificação do mesmo era necessário alterar os dados de entrada, reprocessar e refazer o desenho. A idéia de Computação Gráfica Interativa é de que o usuário pode intervir na construção do desenho e visualizar o resultado imediatamente. Por exemplo, pode-se alterar a posição de uma linha no desenho sem que o mesmo tenha que ser redesenhado. Apenas a linha escolhida é alterada.

Atuando também nesta área de pesquisa vale salientar o trabalho desenvolvido na General Motors, o DAC-I, sistema gráfico criado para emular e melhorar os métodos tradicionais de projeto de automóveis. Também o trabalho dos pesquisadores da Lockheed-Georgia Company, com a aplicação de técnicas gráficas interativas para a definição de formas geométricas, que serviriam como entrada de dados para programas de análise de estruturas pelo Método de Elementos Finitos, e produção de programas perfurados em fitas para máquinas ferramentas de comando numérico.

Seguindo estes esforços pioneiros, muitos pesquisadores de outras indústrias e universidades desenvolveram seus projetos de pesquisa em Computação Gráfica. Cada grupo desenvolveu um software aplicativo próprio para seus computadores e periféricos. Em virtude do alto custo de desenvolvimento do software gráfico, surgiu a necessidade de pacotes gráficos de uso geral, para auxiliar o programador de aplicações a prover recursos gráficos em seus aplicativos.

1.3. Software de apoio à programação

Muitos softwares de apoio à programação foram desenvolvidos, de modo a eliminar a necessidade do programador de aplicações ser também um especialista em Computação Gráfica. Algumas maneiras de se prover estes recursos ao programador foram mais exploradas, entre elas :

- linguagens gráficas.
- extensões às linguagens de programação já existentes, implementando-se recursos gráficos.
- bibliotecas de subrotinas gráficas ou pacotes gráficos de uso geral.
- editores gráficos.

1.3.1. Linguagens gráficas

No início do desenvolvimento de ferramentas gráficas de auxílio à programação, a criação de linguagens gráficas foi bastante pesquisada. Foram desenvolvidas algumas linguagens deste tipo para fins específicos, como por exemplo trabalhos artísticos, animação para vídeo e cinema, etc, para os quais estas linguagens apresentaram-se como uma boa solução. Entretanto, no desenvolvimento de aplicativos na área de engenharia, como por exemplo programas para análise de estruturas, que envolvem muitos cálculos e recursos científicos, estas linguagens tornaram-se insuficientes. Os pesquisadores que voltavam seus esforços para a criação de linguagens gráficas

reconheceram a necessidade dos melhores recursos de programação encontrados nas linguagens padrão. Deste modo muitos deles voltaram seus esforços para a implementação de recursos gráficos nas linguagens já existentes.

1.3.2. Extensão das linguagens de programação

A extensão de linguagens de programação, implementando-se às mesmas recursos gráficos foi, durante muito tempo, a maneira mais procurada para prover recursos gráficos aos programadores de aplicativos. A interface com o programador fica bastante satisfatória, sendo que o compilador se encarrega da checagem de erros, e o código fonte fica bastante claro e elegante. Hoje, a maioria dos compiladores, principalmente para computadores pessoais (PC-XT, AT ou 386), apresentam bons pacotes gráficos embutidos na linguagem. Entretanto, nem sempre o melhor compilador (e a melhor linguagem) para a parte gráfica interativa de um programa é também o melhor para o desenvolvimento do aplicativo em questão. É o caso que ocorre frequentemente com programas de análise de estruturas por Elementos Finitos, geralmente escritos em FORTRAN, cujos pré e pós-processadores para entrada e saída de resultados deveriam ser preferencialmente escritos em linguagens como Pascal ou C, que apresentam excelentes recursos gráficos e fácil interação com o usuário do programa.

1.3.3. Bibliotecas gráficas

Existem hoje disponíveis no mercado vários pacotes gráficos para apoio à programação. Estes pacotes podem ser bibliotecas de subrotinas gráficas fornecidas em código fonte, para serem adicionadas ao fonte do aplicativo, e compiladas por algum compilador específico, ou então bibliotecas de funções gráficas já compiladas, geralmente com versões para diversas linguagens de programação, que podem ser ligadas ao aplicativo, também já compilado, por meio de um ligador (*linker*), quando será gerado o código executável.

Quando as subrotinas são fornecidas em código fonte, tem-se a possibilidade de alteração das mesmas, visando um melhor aproveitamento para o fim específico a que são destinadas. Entretanto, a modificação dos fontes implica em um grande conhecimento não só da estrutura do pacote gráfico, como também de Computação Gráfica, por parte do programador do aplicativo. Além disso, estas bibliotecas são geralmente construídas para um compilador específico, não sendo possível sua utilização com outros compiladores. Um exemplo deste tipo de pacote é o *Graphix Toolbox*, fornecido pela *Borland*, para ser utilizado juntamente com seu compilador *TurboPascal*.

As bibliotecas gráficas para uso geral, fornecidas em código objeto, já compiladas, geralmente não apresentam possibilidade de modificação, ou então é bem restrita. Contudo, são bastante fáceis de se utilizar, devendo o programador de

aplicativos conhecer apenas sua estrutura e a chamada de cada função com seus parâmetros. As bibliotecas deste tipo mais conhecidas são propostas para padronização, no sentido de encontrar uma norma para o interfaceamento gráfico. Hoje são disponíveis versões do PHIGS (*Programmer's Hierarchical Interactive Graphics System*), que é objeto de estudo para padronização em desenvolvimento na ISO e ANSI; o CGI (*Computer Graphics Interface*), também em estudo pela ISO e ANSI; e o GKS (*Graphical Kernel System*), que foi adotado como norma internacional pela ISO em 1985 e como norma americana pela ANSI, também em 1985, e está sendo traduzido para o português pela Comissão de Estudo do GKS "CE-21:203.01" da ABNT, para torná-lo o padrão brasileiro.

1.3.4. Editores gráficos

Os Editores Gráficos são pacotes de software interativos que permitem ao usuário criar desenhos de engenharia com auxílio do computador, visualizar detalhes com funções de *zoom*, e imprimí-los em periféricos de saída gráficos, como impressoras matriciais ou traçadores gráficos (*plotters*). Além disso, podem funcionar como ferramentas para entrada e saída gráfica de dados em programas aplicativos de engenharia, através de uma interface de conversão entre os arquivos do aplicativo e do editor.

Normalmente, os Editores Gráficos são utilizados apenas como ferramentas para desenho. Desta forma, são construídos visando dar ao usuário um grande número de funções e recursos

gráficos, como várias cores ou tipos de linhas, diferentes espessuras de traçado, etc. Assim, torna-se necessário o armazenamento de muitos parâmetros para a definição de cada entidade geométrica definida, o que por sua vez torna o arquivo de dados gráficos bastante grande e complexo. A estrutura ou especificação destes arquivos é feita então de forma a melhor conter todos os parâmetros que definem cada entidade gráfica. Deste modo, cada fabricante de sistemas deste tipo adota sua própria especificação para o arquivo de dados, e esta especificação é feita de modo a permitir uma rápida recuperação dos dados, sem preocupação com a possibilidade de comunicação do sistema com outros programas.

Um exemplo de aplicação de um editor gráfico como ferramenta de apoio à programação esclarece ainda mais sua utilidade na engenharia. Um programa típico para dimensionamento de engrenagens tem, como resultado final, as dimensões da engrenagem calculada. esta tabela de dimensões pode ser convertida para uma tabela de coordenadas, dispostas segundo a especificação de um editor. Com a chamada do editor, pode-se então visualizar numa tela de vídeo a engrenagem calculada, e ainda observar detalhes com funções de *zoom*. Tem-se também a possibilidade de modificar o desenho, ou incluir mais detalhes, utilizando os recursos de edição do editor, e por fim gerar uma cópia do desenho em papel, impressa por um dispositivo de saída gráfica. As funções do editor podem aumentar ainda se estas informações gráficas forem interpretadas, e utilizadas para a geração do programa em lingua-

gem específica que alimenta a máquina de comando numérico, responsável pela usinagem da peça.

1.4. Editor Gráfico para Projeto Mecânico

Analisando as características dos softwares para apoio à programação de aplicativos mecânicos, chega-se a conclusão de que, especialmente para a aplicação específica em projeto mecânico, a saída gráfica gerada em um editor gráfico é mais interessante. Independente da visualização da peça projetada, um editor possui as ferramentas necessárias para que o projetista realize pequenas alterações no desenho da peça, inclusão de detalhes, legendas ou anotações e, principalmente, uma capacidade de reprodução do desenho com total controle sobre a escala de desenho, folhas, tamanho do papel, etc.

Desta forma, pode-se criar um sistema de projeto com capacidade para dimensionamento de uma peça a partir de um programa de cálculo, permitir sua visualização com detalhes, e ainda reproduzir seu desenho em papel adequado, já pronto para a linha de fabricação, tornando o processo de projeto mais rápido e eficiente.

Será então desenvolvido um editor gráfico cuja principal característica é a facilidade de comunicação com outros softwares, permitindo a fácil montagem de seus arquivos a partir de dados gerados por programas aplicativos. Estes arquivos também devem ser facilmente interpretados, de modo a dar

continuidade no processo de fabricação e servir como alimentação para sistemas de manufatura auxiliada por computador *(CAM)*. Além disso, é prevista uma boa interface de saída gráfica em dispositivos gráficos de saída, como traçadores gráficos ou impressoras matriciais.

CAPITULO 2

A ESTRUTURA DO PROGRAMA

2.1. *Introdução*

Um sistema para projeto mecânico auxiliado por computador, do tipo que se pretende construir, deve possuir um editor gráfico como módulo central, e a possibilidade de ligação com os vários módulos de cálculo disponíveis. Pode-se definir três classes de usuários para um sistema deste tipo :

- operadores do sistema (desenhistas ou projetistas).
- programadores de aplicação.
- programadores do sistema.

O operador do sistema deve ter acesso somente às informações que permitem operar qualquer módulo de aplicação ou o editor gráfico. Não exige nenhuma experiência de programação.

O programador de aplicações deve ter as informações necessárias para a ligação de seu aplicativo ao editor gráfico, principalmente quanto às rotinas de gerência e controle do editor. É essencial que o programa de aplicação possa ser construído e integrado ao editor sem que o programador de aplicações tenha que se preocupar com os dados de baixo nível,

detalhes do sistema gráfico ou rotinas de manipulação de periféricos. Modificações do sistema devem ser inclusive evitadas para não permitir a incompatibilidade entre sistemas no futuro.

O editor gráfico deve então permitir o acesso aos dados gerados pelos módulos de aplicativos, a manipulação destes dados sob a forma de desenho, e a saída destes dados na forma de uma cópia impressa, ou um arquivamento em disco para posterior utilização. Para a realização eficiente destas tarefas foram estudadas duas estruturas para o sistema, que são a estrutura do GKS (*Graphical Kernel System*), e uma estrutura alternativa, mais livre. Também quanto à especificação do formato do arquivo de dados ou arquivo gráfico foram analisadas algumas alternativas envolvendo padrões e técnicas de armazenamento.

2.2. O G.K.S.

O *GKS* (*Graphical Kernel System*) [15,16,17,24,25] é um pacote gráfico cuja especificação foi aprovada como padrão internacional pela ISO (*International Standard Organization*) em 1984, e por diversas outras entidades nacionais de padronização, como a DIN e ANSI. A ABNT tem uma comissão trabalhando na tradução da norma para o português, de modo a tornar o *GKS* o padrão gráfico brasileiro.

O pacote gráfico *GKS* é um núcleo de funções gráficas padronizadas que atendem a várias tarefas inerentes a um sistema

gráfico, como a geração, representação e manipulação de figuras (imagens) geométricas e não geométricas (caracteres de texto), o controle funcional de dispositivos de entrada e saída gráfica conectados ao sistema, manipulação dos recursos de entrada do sistema, a estruturação de figuras em partes manipuláveis independentemente, e o armazenamento das informações gráficas para uso posterior.

O pacote é considerado um núcleo de um sistema gráfico sobre o qual sistemas gráficos orientados para determinadas aplicações seriam construídos, sem que o programador destes sistemas tenha a necessidade de conhecimento, a nível de hardware ou instruções de máquina, dos dispositivos gráficos. Este núcleo é acessado por intermédio de funções expressas na linguagem de programação do programa. O esforço de especificação do *GKS* inclui não só a especificação funcional do núcleo, mas também das estruturas e funções do núcleo expressas em diversas linguagens de programação.

O *GKS* posiciona-se entre o programa de aplicação e os os diversos periféricos gráficos, que são normalmente acessados via sistema operacional. Para tornar estes programas independentes das caraterísticas específicas e dos códigos de controle dos periféricos, a especificação *GKS* introduziu uma série de conceitos e modelos. Através destes modelos ficam definidos, para o programa de aplicação, periféricos virtuais que se comportam de uma maneira uniforme. Desta maneira, o programa de aplicação trata

um vídeo da mesma maneira que um traçador gráfico. As especificações completas destes periféricos ficam com o *GKS*.

Os conceitos e modelos introduzidos pela especificação ão *GKS*, de modo a criar um ambiente de programação completamente independente de dispositivos são :

Saída gráfica : as figuras criadas no sistema são formadas por elementos básicos chamados primitivas gráficas de saída. Cada figura possui a si associada uma série de atributos (como cor, tipo de linha, etc) que controlam o aspecto visual de sua exibição.

Sistemas de coordenadas : o *GKS* trabalha com três sistemas de coordenadas. O programa de aplicação expressa as entidades geométricas em um sistema de coordenadas que pode ser cartesiano ou polar. Este sistema representa as coordenadas do mundo real ou coordenadas do usuário. O *GKS* faz a transformação para um sistema de coordenadas neutro, chamadas coordenadas normalizadas, que são definidas sempre no intervalo (0,1). Enfim estão as coordenadas do dispositivo, característica de cada periférico conectado ao sistema. Se faz necessária outra transformação para a representação das entidades geométricas no dispositivo selecionado.

Estação de trabalho : o padrão *GKS* foi concebido de forma a ser aplicável aos diversos tipos de ambientes gráficos existentes, o que é de fundamental importância, tendo em vista a

grande variedade de dispositivos de entrada/saída ofertados atualmente. Assim, o conceito de estação de trabalho, tomado em sua forma mais simples, é tido como uma superfície de visualização e dispositivos lógicos de entrada/saída. Este conceito abstrato de estação de trabalho é descrito funcionalmente através de uma tabela de descrição de dispositivos associada. Por ser um conceito lógico e não físico, o usuário do sistema pode acessar qualquer dispositivo gráfico de maneira uniforme.

Segmentação : as figuras criadas pelos programas de aplicacão podem ser estruturadas em partes que, como uma unidade chamada, podem ser manipuladas independentemente. Cada unidade tem seus próprios atributos. Deste modo, uma saída gráfica pode ser armazenada sob a forma de segmentos, tanto localmente, na própria estação de trabalho em que foi gerada (*WDSS - Workstation Dependent Segment Storage*), ou por meio do armazenador de segmentos independente da estação de trabalho (*WISS - Workstation Independent Segment Storage*), que permite a transferência de segmentos entre as estações de trabalho.

Entradas gráficas : para manter a comunicação do operador da estação de trabalho com o programa aplicativo, o *GKS* prevê, além da entrada de dados geométricos, como um par de coordenadas, outros dispositivos lógicos de entrada que incluem entradas alfanuméricas, de seleção (botões) e de valores reais (alavancas e potenciômetros). Estes tipos de entradas podem ser

utilizáveis de três modos : por amostragem, pedido ou fila de eventos.

Metarquivo (*metafile*) : para evitar a perda das informações geradas ao fim de uma sessão de trabalho, o *GKS* prevê a existência de um metarquivo, um arquivo sequencial de imagens, que permite o armazenamento a longo prazo destas informações e seu posterior uso, seja no mesmo programa ou ainda em outro sistema. Os metarquivos são tratados pelo *GKS* como categorias especiais de estacões de trabalho.

O *GKS* é o único padrão a nível de rotinas gráficas adotado internacionalmente. Existem outras propostas, como o *CORE* e o *PHIGS*, não oficializadas. O *PHIGS* (*Programmers Hierarchical Interactive Graphics System*) tem tido uma aceitação crescente, em virtude de ser tridimensional, ao passo que o *GKS* é limitado em duas dimensões.

2.3. Uma estrutura alternativa

A utilização de uma estrutura seguindo um padrão, como o caso do GKS, trás uma série de vantagens, como a portabilidade do sistema criado e facilidade de comunicação entre programadores que utilizam a mesma estrutura. Entretanto, como o propósito do sistema é rodar em ambiente DOS, em microcomputadores do tipo PC-XT, uma estrutura mais livre pode trazer mais vantagens, como o melhor aproveitamento de todos os recursos que a máquina pode dar.

De [16] tem-se a descrição de uma estrutura modular para sistemas de projeto auxiliado por computador. De uma forma geral, estes sistemas apresentam cinco módulos principais, como mostrado no diagrama da figura 2.1.

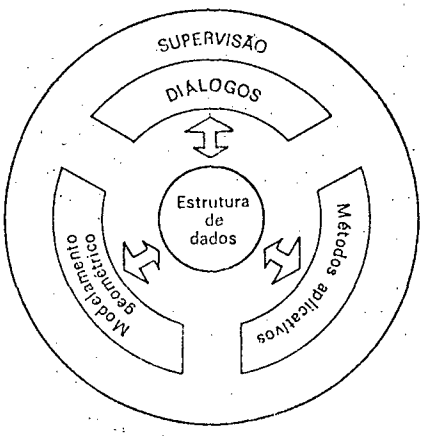


Figura 2.1 - Uma estrutura modular.

- **Diálogos** : é o módulo responsável pela entrada de dados e exibição de resultados, troca de comandos e mensagens entre o sistema e o operador.

- **Modelamento geométrico** : é o módulo que contém as funções de criação e manipulação das entidades geométricas.

- **Métodos aplicativos** : são os programas específicos de aplicação, quer sejam programas de cálculo, análise ou simulação física de comportamentos.

- **Dados** : corresponde a todas as árvores, ponteiros e demais estruturas que devam ser definidas, organizadas e preenchidas, de forma a possibilitar o modelamento dos objetos a serem projetados.

- **Supervisão** : corresponde aos programas de gerenciamento e controle dos demais módulos e da interação com o usuário, de

forma a garantir a integridade do sistema e a sua boa utilização.

2.3.1. Armazenamento de dados

Este tipo de arquitetura para programas de *CAD* não implica na utilização de uma estrutura de dados pré-definida. Deste modo, tem-se liberdade para escolher a que mais convier. A primeira estrutura que se pode pensar para o armazenamento das informações de um desenho é uma lista sequencial, onde os dados são armazenados na ordem em que são introduzidos. Um arquivo no formato ASCII de caracteres, contendo uma lista sequencial, é a forma mais simples de armazenamento, mas que leva grande desvantagem em termos de velocidade de acesso às informações para o arquivo formatado com armazenamento binário.

Numa estrutura de dados tipo lista sequencial, os pontos que possuem duas ou mais linhas a eles conectadas são armazenados mais de uma vez. Deste modo, poderia-se salvar espaço se fossem armazenados os pontos separadamente de suas interconexões, em dois arquivos separados. Este método foi analisado [16], chegando-se às seguintes conclusões :

- produz uma boa redução no volume de dados.
- acesso lento para um volume de dados muito grande.
- complica e dificulta a geração do arquivo gráfico a partir de programas aplicativos.
- inflexível, porque diferentes tipos de dados são difíceis de serem acrescentados.

Outro método utilizado para economizar volume de dados é realizar o armazenamento de "macros" (sequência repetida de dados) apenas uma vez. Os macros podem ser armazenados em um arquivo de "desenho de macros" e referenciado por um código na estrutura de dados, especificando não apenas os macros utilizados, mas sua posição, escala e ângulo de rotação para inserção no desenho. Este método traz, entretanto, algumas desvantagens :

- a edição do conteúdo de um macro se torna difícil quando for posicionado em um desenho.
- o tempo de exibição é um pouco aumentado, pois é necessária a aplicação das transformações (rotação, escalamento) antes de sua exibição.
- a localização de um ponto ou identificação de um elemento dentro do macro requerem rotinas complicadas de busca, pois só há referências ao bloco e não aos elementos que o compõe.

2.3.2. A especificação *IGES*

O *IGES* (*Initial Graphics Exchange Specification*) é uma especificação internacional que define formatos padrão de arquivos para transmissão de informações do banco de dados de um sistema *CAD* para o banco de dados de outro sistema *CAD*, visando a geração de desenhos e outros tipos de representações gráficas.

A especificação *IGES* é utilizada como padrão para transferência de informações gráficas entre sistemas diferentes, para a geração, por um sistema *CAM*, de códigos de comando numérico para a fabricação de uma peça projetada em um sistema *CAD*, ou então como arquivo para armazenamento de dados de sistemas *CAD*. Um sistema que utiliza esta especificação tem muitas facilidades em termos de comunicação com qualquer outro sistema de *CAD* ou sistemas *CAM*, uma vez que estes sistemas apresentam sempre pré e pós-processadores *IGES*.

Um arquivo no formato especificado pelo *IGES* é uma lista de entidades que visam a definição de um produto. Estas entidades estão divididas em três áreas :

- entidades geométricas : são em número de 20, indo desde elementos simples como linha e arco de círculo, até elementos complexos, de uso mais restrito, como B-splines racionais.
- entidades de anotação : correspondem à informação usada pelo projetista para descrever como o produto deve ser fabricado. Estas entidades são dimensões lineares, radiais, angulares, linhas de centro, etc.
- entidades de estrutura : são utilizadas para comunicar a estrutura de banco de dados de um sistema *CAD/CAM* para outro. Isso assegura que informações lógicas, como a associação de elementos para formar uma peça, não se percam numa transferência via *IGES*.

As implementações atuais disponíveis do *IGES* utilizam um arquivo com formato ASCII normal, com 80 colunas por registro. Este arquivo contêm cinco tipos de dados, que são :

- inteiros
- valores de ponto flutuante
- cadeias de caracteres
- apontadores
- comandos de linguagem

A estrutura de dados do *IGES* é dividida em cinco seções principais.

A seção de início provê um prólogo para o arquivo.

A seção global contêm informações que descrevem o pré-processador e informações necessárias para o pós-processador manipular o arquivo *IGES*, tais como precisão, valores máximos e mínimos do desenho, escala, unidades, etc.

A seção de diretório contêm as especificações de cada entidade do desenho, ou seja, seu número, um apontador que aponta para o local onde estão seus parâmetros, e seus atributos (cor, tipo de linha, etc).

A seção de dados de parâmetros contêm os parâmetros das entidades *IGES*. O número de parâmetros depende de cada entidade, sendo que são armazenados todos os pontos e informacões específicas das entidades que constam da seção de diretório.

A seção de terminação é formada por apenas um registro que indica o fim do arquivo.

2.4. A escolha da estrutura

Na escolha da estrutura adotada para a construção do editor gráfico, será tomado como fator mais importante a estrutura de armazenamento de dados, ou seja, o formato do arquivo gráfico. É por meio deste arquivo que o editor se comunicará com os programas aplicativos de projeto mecânico, e assim constituir o sistema integrado de projeto de componentes mecânicos.

Outro fator que deve ser levado em consideração é o fato ito de o editor estar sendo desenvolvido em ambiente acadêmico. Neste meio estão sempre surgindo novas idéias, novos métodos e algoritmos para tornar rotinas gráficas mais eficientes. Deste modo, é importante construir um programa aberto e modular, de forma que novos algoritmos possam ser facilmente implementados e testados, e mesmo que pessoas diferentes possam dar manutenção e principalmente continuidade ao trabalho.

Também para futuras e possíveis utilizações do editor, como por exemplo pré e pós-processadores para programas de análise por Elementos Finitos, que necessitam de alterações na estrutura do programa e também em sua estrutura de dados, é interessante que o programa permita estas alterações sem que um esforço muito grande deva ser dispendido. No caso de programas

de análise, ter-se-ia que alterar a estrutura de dados para que fossem armazenados, além da geometria do objeto de estudo, de suas vinculações, esforços, conectividade dos elementos, etc.

Resumindo os requisitos necessários do editor, temos que tanto sua estrutura do programa quanto a estrutura de dados devem ser o mais flexível possível, de modo a atender as modificações que tornariam seu campo de aplicações mais amplo. Esta flexibilidade quanto as alteracões deve estar necessariamente aliada a uma modularidade que permita uma manutenção e novas implementações de maneira simples e eficiente.

Para tentar satisfazer os requisitos de construção do editor, foi estudado o *GKS*. O *GKS* é um núcleo gráfico que contêm funções para a criação e manipulação de entidades geométricas, uma estrutura de dados para armazenamento e processamento de imagens geradas, e protocolos de comunicação para os diversos periféricos gráficos utilizados como entrada/saída. Deste modo, a criação do editor ficaria limitada a confecção de um módulo gerenciador, que faria as chamadas ãs rotinas do *GKS*, a partir da interpretação de comandos do usuário. Por ser também um padrão gráfico internacional, torna-se mais fácil para que programadores diferentes possam realizar a manutenção do programa, ou mesmo a implementação de novas funções.

A estrutura do *GKS* apresenta, entretanto, vários inconvenientes. Inicialmente, a única versão padronizada do *GKS*

disponível é uma ligação para a linguagem de programação FORTRAN. O FORTRAN é uma linguagem que apresenta ótimos recursos para a programação científica, mas com pouquíssimos recursos na parte de interface com o usuário. Como o editor é um programa essencialmente interativo, que depende muito de uma boa conversação com o usuário, é mais recomendado o uso de linguagens como o Pascal ou C, que fornecem mais recursos neste sentido.

A estrutura de segmentos do *GKS* o torna rígido e inflexível para certos tipos de mudanças. A estrutura do editor ficaria então amarrada a este tipo de estrutura, que permite muito pouca liberdade para novas aplicações. A estrutura de dados também é inflexível, baseada nos conceitos de metarquivo e segmentação.

Como o editor irá rodar em máquinas tipo PC-XT e AT, a elaboração de algoritmos específicos para este hardware certamente fornecerá soluções mais eficientes que as rotinas gráficas do *GKS*, que são formuladas de uma maneira bastante geral, para atenderem o requisito de independência do hardware. Devido à performance dos microcomputadores não ser muito boa, a eficiência ou velocidade das rotinas gráficas é essencial para um bom desempenho do operador sobre o sistema, pois se o sistema for muito lento em suas respostas, o trabalho se tornará muito cansativo. Os códigos executáveis gerados sem o *GKS* também serão menores, o que poupará espaço em disco e principalmente em memória, que é bem limitada neste hardware.

Por todos estes motivos, optou-se por abandonar o *GKS* como ferramenta para o desenvolvimento do editor, e utilizar uma biblioteca gráfica em código fonte, juntamente com a linguagem de programacão Pascal. Como o Pascal é uma linguagem muito difundida em microcomputadores, encontram-se hoje várias bibliotecas de funções e rotinas para os mais diversos fins. No caso específico de rotinas gráficas, tem-se o *Graphix Toolbox*, pacote de rotinas fornecidas em código fonte para serem compiladas pelo compilador Turbo Pascal. Como são fornecidas em código fonte, podem ser alteradas e adaptadas para utilizações mais específicas, ou mesmo substituídas por algoritmos mais eficientes. Fica a cargo do programador a definição da estrutura do programa, uma vez que o *Graphix* não possui nenhuma estrutura pré-definida.

Será construída uma estrutura modular, conforme proposto por [16], sendo os módulos principais : módulo de gerenciamento, que supervisiona o fluxo de dados e informações entre os demais módulos; módulo de interface com o usuário, responsável pela entrada de dados e comandos, troca de mensagens e manipulação de menus; módulo de modelamento geométrico, onde estão as rotinas de criação e manipulação das entidades geométricas; e o módulo de dados, responsável pelo armazenamento dos dados gráficos tanto na memória quanto em disco. Os módulos de aplicativos mecânicos serão incorporados ao editor mais tarde, e se comunicarão com o mesmo através do arquivo gráfico.

O formato escolhido para armazenamento dos dados em disco sco é a lista sequencial, em virtude de sua simplicidade e facilidade de criação, o que vem de encontro com o principal objetivo do editor, que é a comunicação com outros programas. A especificação *IGES* é muito completa, de forma a garantir a troca de informações entre qualquer sistema gráfico. Por isso mesmo, é muito complexa, e de difícil criação a partir de programas aplicativos. Os arquivos se tornam muito extensos, dificultando e tornando mais lenta sua interpretação e leitura do disco.

A opção de utilizar o arquivo de armazenamento de pontos separados de suas interconexões [16] foi também rejeitada pelo principal motivo de dificultar muito a interpretação e geração do arquivo gráfico a partir dos programas de aplicação. A idéia do armazenamento de sequências de informacões repetidas em macros foi adotada, com algumas modificações. Os macros são no editor chamados de blocos, que são armazenados em disco. Quando chamados em um desenho, todos seus elementos são adicionados ao desenho individualmente, não sendo gravado no arquivo gráfico do desenho nenhuma informação referente ao bloco chamado. Apenas seus elementos são transferidos para o desenho. Deste modo não há economia de espaço para armazenamento, mas a simplicidade e praticidade do arquivo são mantidas. A grande vantagem do uso dos blocos é a confecção de bibliotecas de elementos mais usados, que podem ser referenciados pelos programas aplicativos, tornando bem mais fácil e rápida a elaboração de interfaces entre os programas aplicativos e o editor.

Ficou definida então uma estrutura modular para o programa, com base na biblioteca de rotinas gráficas *Graphix Toolbox*, utilizada juntamente com o compilador TurboPascal. O armazenamento de dados é feito num arquivo em lista sequencial, gravado no padrão ASCII de caracteres, que é a forma mais simples de armazenamento.

CAPITULO 3

A ARQUITETURA DO SISTEMA

3.1. A interface

A comunicação dos módulos de projeto mecânico com o Editor será feita sempre por meio do arquivo de dados gráficos, gravado em disco. A especificação completa deste arquivo está no capítulo 4 - O Arquivo Gráfico. Assim sendo, a saída de resultados de um programa de cálculo e dimensionamento deve sofrer uma transformação, a ser feita por uma interface, de modo a gerar aquele arquivo. O Editor pode então ser chamado, e por meio de seus recursos, ler o arquivo gráfico para a memória, interpretá-lo e mostrá-lo no vídeo, permitir modificações no desenho e sua impressão em periféricos gráficos de saída.

Uma vez que o arquivo gráfico é o modo de comunicação entre os programas de projeto com o Editor, pode-se construir vários destes módulos ou programas, em qualquer linguagem de programação. Há inclusive a possibilidade de se desenvolver e rodar o programa de projeto em máquinas mais rápidas, como mainframes ou minicomputadores, desde que a saída gerada esteja de acordo com o padrão do Editor, e se tenha condições de passar

este arquivo para o ambiente DOS em microcomputadores PC, onde o Editor é executado.

Assim, pode-se não só desenvolver o programa de projeto mecânico na linguagem que mais convêm para cada caso, como aproveitar os programas já desenvolvidos em outras linguagens que não o Pascal, ou mesmo desenvolvidos em outro tipo de máquina. Como não há restrições quanto a linguagem, basta acrescentar a estes programas uma interface de conversão dos seus dados de saída para o formato do arquivo gráfico do Editor.

Uma outra alternativa de utilização do Editor é a construção de sistemas completos, acoplando-se ao fonte do Editor alguns módulos para projeto mecânico. Deste modo apenas um executável é criado, tornando a comunicação e interação entre os programas de cálculo e os recursos gráficos mais rápida e cômoda. Esta alternativa implica, porém, que os módulos de projeto sejam escritos em Pascal, e que o programador de aplicações tenha um conhecimento maior da estrutura do código fonte do Editor.

A implementação de um módulo de projeto mecânico no Editor é bastante simples. Com o compilador TurboPascal versão 4.0 tem-se o conceito de unidades de programa (*Turbo Pascal Units - ".TPU"*). Estas unidades podem ser compiladas independentemente do resto do programa, devendo-se apenas colocar as devidas chamadas do módulo instalado, bem como a opção no menu principal. A comunicação continua a ser feita pelo arquivo gráfico em disco.

3.2. O Editor

O Editor apresenta dois ambientes de trabalho, basicamente um gráfico e outro não gráfico. Quando o Editor é carregado, entra-se inicialmente no ambiente não gráfico, que mostra o menu principal de opções. Deste menu pode-se encerrar o programa e voltar ã tela do DOS, entrar no ambiente gráfico editando ou criando um desenho, fazer algumas manipulações de arquivos em disco, geralmente para manutenção, ou chamar o módulo de projeto mecânico, de onde tem-se acesso aos programas de projeto que foram incluídos na versão do Editor que se está rodando.

A opção "utilidades" permite o acesso a funções como deletar ou renomear arquivos, listagem do diretório, etc. Estas funções são incluídas para que se possa dar uma certa manutenção no disco de trabalho sem ter que encerrar o programa e carregá-lo novamente após a manutenção.

A outra opção não gráfica é justamente a chamada do menu de projeto mecânico. Dependendo dos módulos para projeto instalados em cada versão, tem-se acesso a cada um deles por meio de um menu secundário, chamado pela opção "projeto mecânico". Cada módulo pode ser chamado e rodado, gerando assim um arquivo gravado em disco. Encerrando-se o trabalho no menu de projeto mecânico, volta-se ao menu principal, donde pode-se editar o desenho gerado pelo programa de cálculo, por meio do

comando de edição. A partir daí entra-se em ambiente gráfico, e tem-se acesso aos recursos de zoom, manipulação, modificação e impressão do Editor, mostrando no vídeo o desenho da peça recém dimensionada.

3.3. O ambiente gráfico

A entrada no ambiente gráfico do Editor é feita a partir do menu principal. Pode-se iniciar um novo desenho com a opção "criar desenho", ou chamar um desenho já existente com a opção "editar". O vídeo é colocado em modo gráfico, e aparece a tela de desenho, delimitada por uma moldura, sendo que a última linha da tela é reservada para entrada de dados e mensagens do programa.

Neste ambiente tem-se acesso a um conjunto de funções e comandos necessários para a criação e alteração de desenhos técnicos. As funções ou comandos podem ser chamadas pela digitação de seu nome na linha de comando, ou através dos cinco menus cíclicos do Editor. Estas funções e comandos podem ser classificadas em grupos, que são :

- função de posicionamento.
- função de identificação e seleção.
- comandos de construção.
- comandos de manipulação de elementos.
- comandos de manipulação de blocos.
- comandos de controle de parâmetros.
- comandos de visualização.

- comandos de apoio.
- comandos de reprodução.

3.3.1. Função de posicionamento

É utilizada para indicar um ponto no desenho. Se faz necessária quando da criação de novos elementos, ou manipulação de elementos ou blocos. A indicação de um ponto pode ser feita das seguintes maneiras :

- posicionamento do cursor na tela.
- entrada de coordenadas (x, y) pelo teclado.
- entrada de coordenadas relativas ao último ponto dado (dx, dy).
- entrada de coordenadas relativas ao último ponto dado (ângulo e distância).

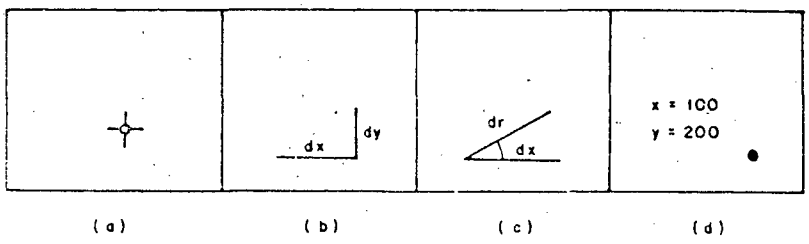


Figura 3.1 - As maneiras de posicionamento de um ponto

a. posicionamento do cursor
b. coordenadas relativas (dx, dy)
c. coordenadas relativas com raio e ângulo
d. entrada pelo teclado

3.3.2. Função de identificação

Esta função é utilizada para a seleção, no desenho, de um elemento para operações de manipulação. As funções de manipulação em geral fazem uso da função de identifição, pois necessitam de um elemento como parâmetro principal. No Editor, a identificação é feita por posicionamento do cursor sobre o elemento desejado. O mecanismo para a identificação utilizado é baseado no retângulo envolvente de cada elemento.

O retângulo envolvente é o menor retângulo de lados horizontais e verticais que envolve completamente o elemento. É definido automaticamente pelo Editor quando da criação de cada entidade gráfica, e é uma figura imaginária, pois não aparece na tela. Para identificar um elemento, o projetista deve

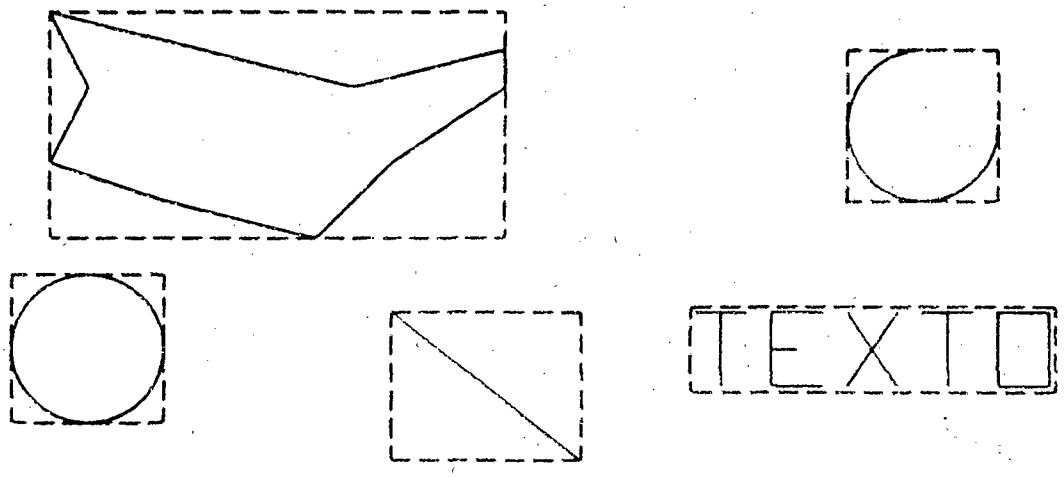


Figura 3.2 - O retângulo envolvente de cada elemento.

posicionar o cursor dentro deste retângulo imaginário, que envolve o elemento desejado. Na figura 3.2 são mostrados alguns

exemplos de retângulos envolventes para algumas figuras bastante comuns, onde o mesmo aparece em linhas tracejadas.

3.3.3. Comandos de construção

São comandos que adicionam uma nova entidade geométrica ao arquivo gráfico. Estas entidades, chamadas primitivas geométricas, são os elementos básicos a partir dos quais são criados desenhos mais complexos. No Editor são disponíveis os seguintes elementos :

- linha.
- linha poligonal.
- polígono.
- retângulo.
- círculo
- arco de circunferência.
- texto

Linha : O elemento *linha* é um segmento de reta definido pelos seus dois pontos extremos. A indicação dos pontos é feita com auxílio da função de posicionamento.

Linha poligonal : A *linha poligonal* é uma sequência de segmentos de reta unidos, formando um polígono aberto. A entrada de dados para a criação de uma *linha poligonal* é a mesma da *linha*. As diferenças estão na forma de armazenamento dos dados no arquivo gráfico, e a forma com que os elementos podem ser manipulados. Uma sequência de segmentos de reta construídos com o comando

linha é mais flexível quanto a manipulação, pois cada segmento, por ser um elemento, pode ser manipulado separadamente. Entretanto, há casos em que se deseja manipular a sequência toda de uma vez, ocasião em que a *linha poligonal* deve ser preferida. Por questão de economia de espaço, também é mais recomendada a *linha poligonal*, uma vez que cada ponto da sequência é gravado no arquivo gráfico apenas uma vez. No caso de uma sequência de *linhas*, cada ponto é armazenado duas vezes, uma vez para cada linha adjacente.

Polígono : É uma sequência de segmentos de reta fechada. Os dados são armazenados da mesma forma que na *linha poligonal*, sendo que, na sua representação em tela, o último ponto é automaticamente ligado ao primeiro, de forma a fechar a cadeia e formar o *polígono*.

Retângulo : É um caso especial de *polígono*. Por ser uma figura de larga utilização, foi adicionado um comando para criar um *polígono* de lados horizontais e verticais, de modo que possa ser definido por apenas dois pontos. Estes dois pontos são seu canto inferior esquerdo e superior direito, formando uma das diagonais do *retângulo*. No arquivo gráfico são armazenados os quatro pontos, de modo que o elemento possa ser rotacionado (quando seus lados deixam de ser horizontais e verticais).

Círculo : Um *círculo* pode ser definido por seu centro e raio. Estas são as informações gravadas no arquivo gráfico. Para

facilitar o trabalho do projetista, foram implementados no Editor três maneiras diferentes de se criar um *círculo* :

- pela definição do centro e raio.
- pela definição do diâmetro (dois pontos).
- passante por três pontos.

Arco : O *arco* de circunferência é definido por seu centro, raio, ângulo de início e ângulo final. Os ângulos são definidos em relação a horizontal, no sentido anti-horário. Da mesma forma que para o *círculo*, foram previstas maneiras diferentes para a definição de um *arco* :

- pela definição do centro e dois pontos, sendo que o primeiro define o raio e ângulo inicial, e o segundo define o ângulo final.
- passante por três pontos, sendo que o primeiro e último pontos definem os ângulos inicial e final, respectivamente.

Texto : O elemento de *texto* é uma entidade não gráfica muito utilizada em desenhos técnicos, pois se faz necessário para a confecção de legendas, cotagem, listas de material, etc. É formado por uma cadeia de até 70 caracteres, um ponto que define sua posição no desenho, a inclinação (direção de escrita), e o tamanho do fonte, definido pela altura da letra. Pode-se escrever em qualquer tamanho e em qualquer direção.

3.3.4. Comandos de manipulação

São comandos utilizados para a manipulação das entidades geométricas previamente definidas, alterando seus parâmetros ou mesmo eliminando do arquivo gráfico os dados referentes ao elemento selecionado. Estes comandos implicam no uso das funções de posicionamento e identificação para a definição de seus parâmetros. Foram implementados no Editor os seguintes comandos de manipulação :

Eliminar : Apaga o elemento selecionado do desenho, e remove seus dados do arquivo gráfico em memória para um arquivo chamado "arquivo morto". Os elementos apagados por engano podem então ser recuperados. Há a necessidade de limpar o "arquivo morto" periodicamente, para liberar espaço em memória, eliminando definitivamente os elementos do "arquivo morto".

Mover : Realiza uma translação do elemento selecionado. São utilizadas as funções de identificação e posicionamento para, respectivamente, selecionar o elemento e indicar no desenho sua nova posição.

Copiar : É uma operação semelhante a *mover*, sendo que o elemento na posição original não é apagado. Os seus dados são copiados e alterados para a nova posição, e um novo elemento é inserido no desenho, armazenando as informações no arquivo gráfico.

Girar : Realiza uma rotação do elemento identificado. Seus parâmetros são a identificação do elemento, o posicionamento de um ponto que será o centro de rotação, e o ângulo de rotação. Os dados do elemento são alterados no arquivo gráfico e o desenho é rotacionado para a nova posição.

Escalar : Permite a ampliação ou redução de um elemento no desenho, em relação a um ponto dado. É necessária a seleção do elemento, a indicação do ponto de referência, e o fator de escala para a transformação.

3.3.5. Comandos de manipulação de blocos

São comandos com função semelhante aos comandos de manipulação de elementos, agora com capacidade de manipular um conjunto de elementos previamente selecionados, numa só operação. A seleção do conjunto de elementos é feita por meio de uma janela na tela de desenho. A janela é definida por dois pontos que formam uma de suas diagonais, e só são incluídos os elementos que estão completamente no interior da janela. Tendo sido definido um bloco, pode-se então realizar operações como eliminar, mover, copiar, rotacionar e escalar sobre os elementos do mesmo.

Existem ainda dois outros comandos para a manipulação de blocos que são de grande importância para o projeto mecânico. São os comandos que permitem gravar e recuperar um bloco do disco. Desta forma, pode-se desenhar um componente de um

projeto, como um parafuso por exemplo, e armazená-lo em disco. Cada vez que houver necessidade de um parafuso semelhante no desenho, pode-se recuperar o bloco do disco, em qualquer lugar do desenho, e com qualquer escala e ângulo.

Estes recursos permitem a construção de bibliotecas de símbolos e componentes, de modo a facilitar muito o trabalho do projetista. Estes blocos também podem ser utilizados quando há o interfaceamento de um programa de projeto mecânico com o Editor. A interface se torna mais fácil se um grande número de partes já está desenhada, bastando posicioná-las no desenho com a escala apropriada.

3.3.6. Comandos de controle de parâmetros

Alguns atributos dos elementos criados pelo Editor, como tipo de linha ou folha de desenho, são parâmetros que possuem sempre um valor corrente, ou seja, cada elemento criado irá associar o atributo configurado naquele momento. A partir da alteração do valor destes parâmetros, cada novo elemento criado receberá o novo valor para seu atributo, sendo que os elementos criados anteriormente à modificação não são alterados.

Tipo de linha : Com exceção do elemento de texto, todos as entidades geométricas possuem o atributo *tipo de linha*. Para alterar o tipo de linha ativo ou corrente utiliza-se o comando correspondente (*tipo/lin*). O Editor conta com cinco tipos de linhas, como mostrado na figura 3.3.

CODIGO	FORMA	TIPO
0	———————	contínuo
1	. . . . .	pontilhado
2	- - - - - - - -	tracejado curto
3	— — — — —	tracejado longo
4	—·—·—·—·—	traço ponto

Figura 3.3 - Os tipos de linha.

Folha : A folha ou nível de desenho é um recurso que permite ao usuário a elaboração de um desenho bem organizado, que também pode aumentar a velocidade de trabalho do sistema. Cada folha de desenho é como se fosse uma folha de papel vegetal. Cada etapa do desenho pode ser feita em uma folha, e depois visualizada separadamente, em conjunto com outras partes ou mesmo formando a imagem do conjunto completo. Por exemplo, no projeto de uma casa pode-se ter a planta numa folha, a instalação elétrica em outra e a instalação hidráulica numa terceira. Durante o projeto pode-se trabalhar com cada planta específica, e fazer a verificação de interferências com o desenho completo, tornando todas as folhas visíveis.

No Editor, existe sempre uma folha ativa, na qual são inseridos os elementos a medida que vão sendo criados. O comando *folha* permite a troca da folha ativa. Os elementos criados anteriormente ao comando continuam na folha em que foram criados. Somente os elementos criados a partir daquele momento farão parte da nova folha. A folha ativa ao ser inicializado o sistema é a "1". O Editor suporta até 20 folhas. Para permitir

folha ativa nunca pode estar "não visível". Cada comando folha dado torna a nova folha escolhida automaticamente visível.

Quando se está elaborando o projeto de uma parte específica do conjunto, não havendo então a necessidade de visualização do conjunto completo, pode-se tornar todas as folhas não visíveis, ficando somente a folha ativa na tela. Com isto ganha-se tempo, pois a cada operação em que uma atualizacão do desenho é necessária, somente aqueles elementos pertencentes ã folha visível serão redesenhados.

O comando *verfolha* permite o controle sobre quais as folhas de desenho que devem ser desenhadas na tela. A folha ativa, escolhida pelo comando folha é sempre visível, não sendo possível desativar sua visualização pelo comando verfolha.

Grade : A *grade* é uma malha de pontos desenhados na tela, com função de auxiliar o projetista. É como se a tela fosse um papel milimetrado, mas com o espaçamento entre os pontos controlado pelo usuário. Desta forma, pode-se escolher a unidade de grade mais conveniente para cada desenho, ou para cada etapa do desenho. A *grade* dã ao usuário uma boa idéia das dimensões e distâncias entre as entidades geométricas na tela. Os pontos da *grade* são apenas referências, não fazendo parte do desenho. Desta forma, como não constam no arquivo gráfico, nunca serão impressos.

Bloqueio de grade : Este comando faz um "bloqueio" do passo do cursor sobre os pontos da grade. Desta forma, a movimentação do cursor se faz somente sobre os pontos da grade, e a entrada de dados pelo posicionamento do cursor se torna bastante precisa. A grade aqui referida não é, necessariamente, a malha de pontos mostrada pelo comando *grade*. O espaçamento entre os pontos para o *bloqueio de grade* é independente da unidade de *grade*, e os pontos da grade não precisam estar visíveis (ou seja, a grade não precisa estar ativa) para que o bloqueio de grade funcione.

Com o *bloqueio de grade* pode-se entrar com pontos com muita rapidez e principalmente precisão, uma vez que o projetista tem o controle sobre a posição dos pontos sobre os quais o cursor se movimenta. A entrada de dados por valor, pelo teclado, não sofre a ação do *bloqueio de grade*, podendo ser usado quando tem-se uma coordenada que esteja fora dos pontos de bloqueio, e o cursor não acesse aquele ponto.

3.3.7. Comandos de visualização

São comandos que não tem efeito sobre o banco de dados, ou seja, o arquivo gráfico. Apenas a porção do desenho visualizada na tela é alterada. O comando principal de controle de visualização é o *zoom*, sendo que o comando *atualiza* é apenas auxiliar.

Atualiza : Durante o processo de edição ou criação de um desenho, ocorrem ocasiões em que o desenho fica "sujo". Quando

vários elementos estão sobrepostos e um deles é apagado, as intersecções das linhas restantes com as linhas apagadas produzem descontinuidades no desenho. Algumas operações de manipulação podem também produzir imperfeições no desenho. Estas falhas do desenho, entretanto, não constam do arquivo gráfico. São simplemente falhas de representação na tela, que podem ser corrigidas refazendo-se o desenho todo. É esse o propósito do comando *atualiza*.

Zoom : O comando zoom engloba todas as operações de ampliação, redução, enquadramento ou vistas do sistema, permitindo o controle sobre a janela de visão corrente. Este comando possui três opções, a seguir :

- **Zoom janela** : A opção janela do comando zoom permite ao projetista estabelecer uma nova janela de visualização (ou porção de desenho que aparece na tela), indicando dois pontos que definem uma de suas diagonais. Este comando facilita operações de aproximação, ou seja, quando se quer visualizar com detalhes uma parte específica do desenho. Os dois pontos pedidos como entrada definem um retângulo, que indica os limites da janela desejada. A atualização do desenho é automática.

- **Zoom escala** : A opção escala permite realizar um escalamento na janela de visão corrente. Um fator de escala menor que um corresponde a ação de afastar-se da folha de desenho. A área visualizada aumenta, com uma consequente perda de detalhes. Um fator de escala maior que um corresponde a uma aproximação à folha de desenho. A área visualizada diminui, mas tem-se um detalhamento maior do desenho, facilitando a criação

ou edição de pequenas partes do mesmo. Além do fator de escala é pedido também um ponto. Este ponto indicará a centragem do desenho na janela de visão, ou seja, qual a parte do desenho que estará visível. O ponto dado será o centro da janela. O comando *zoom escala* pode também ser usado para deslocar a janela de visão por sobre o desenho. Com um fator de escala unitário, utiliza-se o ponto de centro para indicar a posição da janela de visão. A atualização é automática.

- **Zoom tudo** : A opção zoom tudo tem a função de ajustar o fator de escala e a centragem da janela de visão, de modo que seja a menor janela que contenha o desenho todo. Desta maneira pode-se atualizar a janela de visão para uma vista geral do desenho, sem se importar com escala ou posição da janela anterior. A atualização do desenho é automática.

3.3.8. Comandos de apoio

São comandos de operação sobre arquivos em disco, ou seja, permitem a gravação do arquivo gráfico da memória para o disco, bem como a saída do ambiente gráfico voltando ao menu principal. As funções de apoio implementadas no Editor são :

Limpar : Apaga o arquivo gráfico da memória e limpa a tela gráfica, abandonando todo o desenho feito até aquele momento.

Salvar : Grava em disco o arquivo gráfico do desenho contido na memória. Quando são feitas alterações no desenho é necessário um

novo comando *salvar*, para que as alterações sejam registradas no disco e possam ser recuperadas posteriormente.

Encerrar : Este comando sai do ambiente gráfico, indo para o menu principal, mantendo o desenho editado na memória. Desta forma pode-se realizar algumas operações a partir do menu principal, e voltar ao ambiente gráfico com o desenho ainda na tela e na memória.

3.3.9. Comandos de reprodução

Os comandos de reprodução permitem a saída em papel de desenhos do Editor, por meio de dispositivos gráficos como impressoras matriciais e traçadores gráficos ("plotters"). Estes comandos interpretam o arquivo gráfico e fazem uma conversão segundo o protocolo de comunicação do dispositivo de saída utilizado.

Por meio dos comandos de reprodução tem-se o controle sobre a porção do desenho que será impressa, e o tamanho da área de papel utilizada para a impressão. Desta forma, o projetista determina a escala com que o desenho será impresso, tendo algumas facilidades, como o ajuste da escala para um tamanho de papel determinado. Pode-se também controlar quais as folhas de desenho que serão impressas. Este recurso é importante para a diferenciação dos tipos de linha em traçadores gráficos de uma só pena.

Mais detalhes sobre os algoritmos utilizados na elaboração das interfaces para os periféricos de saída gráficos podem ser encontrados no Apêndice 3 - "Protocolos de Comunicação".

CAPITULO 4

O ARQUIVO GRAFICO

4.1. Introdução

O arquivo gráfico é o arquivo de dados que contém todas as informações necessárias para a completa definição de um desenho. A partir de um arquivo assim construído, pode-se gravar o desenho codificado em disco, e recuperá-lo mais tarde para possíveis modificações (edição), ou para obtenção de cópias em papel.

4.2. Armazenamento de informações

Existem várias maneiras de se guardar as informações de um desenho montado em um sistema gráfico computadorizado. Uma delas é a de transferir todo o conteúdo da memória de vídeo para uma variável, e salvar esta variável em disco. A tela dos micro-computadores padrão IBM-PC, por exemplo, geralmente equipados com uma placa CGA (Computer Graphics Adapter), possui 640 pontos na horizontal e 200 na vertical. Tem-se, portanto, um total de 128000 pontos a serem armazenados. Nestes tipos de vídeo, o trabalho em alta resolução permite apenas uma cor na tela, sobre o fundo escuro. Desta forma, cada ponto da tela,

ou *pixel*, necessita apenas da informação se o mesmo está aceso ou apagado. Esta informação (0 ou 1) ocupa um bit da memória. Portanto, os 128000 pontos da tela podem ser armazenados em 16000 bytes (1 byte = 8 bits), ou 16 Kbytes. Este é o tamanho que deve-se dimensionar a variável que conterá as informações da tela. Este método é bastante rápido, porém não atende aos propósitos do sistema. Com informações apenas sobre os pontos da tela (aceso ou apagado), torna-se praticamente impossível a criação deste arquivo por outros programas, com a finalidade de edição no Editor Gráfico. A manipulação dos elementos que compõe o desenho é também impossível, pois não há condições de identificação de um elemento. Além disto, o tamanho e a resolução do desenho ficam limitados pelo tamanho (número de pontos) e resolução da tela. Devido a estes problemas, um outro método de armazenamento das informações de um desenho foi preferido.

4.3. Estrutura do arquivo

Cada desenho é construído a partir das primitivas geométricas básicas, como linha, círculo, arco, etc. Cada primitiva, ou elemento, como serão daqui por diante denominadas, é adicionada a uma lista na memória do computador quando da sua criação. Devido ao fato dos microcomputadores apresentarem configurações de hardware diferentes, principalmente quanto ao tamanho da memória principal (RAM), optou-se por uma alocação dinâmica da memória. Foi utilizada a estrutura de apontadores que oferece a linguagem Pascal, organizada em forma de lista. Desta maneira,

o desenho na memória está sob a forma de uma grande lista de parâmetros que definem os diversos elementos que formam o desenho. Esta lista pode ser copiada para o disco numa operação de gravação, ou lida de um disco para modificações, plotagem ou visualização, com uma operação de edição.

Nesta lista estão algumas informações iniciais a respeito de detalhes do desenho, como nome do arquivo, fator de escala que o desenho foi gravado ou coordenadas do ponto que será centrado na tela quando da edição. Após estes dados iniciais, segue a lista dos elementos que constituem o desenho. Esta lista é formada por uma série de parâmetros. Estes parâmetros podem ser um código de identificação do elemento, informando se o mesmo é um arco, círculo, linha, etc; alguns atributos do elemento, como tipo de linha (contínua, traço-ponto, pontilhada, etc) ou folha (nível) de desenho a que o elemento pertence; quatro coordenadas que definem o menor retângulo que envolve o elemento, usado para a identificação do elemento; e finalmente os parâmetros que definem o elemento, que são coordenadas de pontos, ângulos, raios, etc.

Com este tipo de estrutura, pode-se facilmente percorrer a lista a procura de um elemento, que será identificado com auxílio do retângulo envolvente comentado acima. O elemento modificado pode sofrer todas as operações de manipulação, como translação, rotação, etc, ou até mesmo ser eliminado do arquivo gráfico. Estas operações são facilmente executadas, pois basta operar sobre as coordenadas definidoras do elemento.

A modificação se torna permanente se o arquivo for gravado em disco.

4.4. *A especificação do arquivo*

Como a estrutura em lista é essencialmente sequencial, a organização do arquivo em disco segue o mesmo princípio. O arquivo em disco é um arquivo tipo texto, gravado segundo o padrão ASCII de caracteres, e tem acesso exclusivamente sequencial. Desta maneira, pode-se editar o arquivo com qualquer processador de texto, alterá-lo ou mesmo criá-lo para posterior edição gráfica. A interpretação do arquivo por parte do usuário também é muito fácil devido a estas características, sendo possível para o mesmo a elaboração de uma interface que converta a saída numérica de um programa para um arquivo capaz de ser editado no Editor Gráfico.

A grande maioria dos sistemas de CAD se utiliza de uma estrutura semelhante para a gravação do arquivo gráfico. Alguns sistemas gravam em binário, o que torna o arquivo de difícil interpretação. Além disto, a maior parte dos sistemas se utiliza de uma padronização própria do arquivo gráfico. Isto gera problemas para a comunicação entre dois sistemas, sendo necessárias duas interfaces para conversão de um padrão no outro e vice-versa.

Existem já algumas tentativas de padronização do arquivo gráfico, como é o caso da especificação IGES (Initial Graphics

Exchange Specification) [16,26]. A especificação IGES é, entretanto, bastante completa e geral, o que a torna de difícil manipulação. A grande maioria dos fabricantes de sistemas CAD optou por trabalhar com um padrão próprio, e oferecer juntamente com o sistema duas interfaces de conversão, uma do seu sistema para o IGES e outra do IGES para o seu sistema. Desta maneira, o IGES é hoje um padrão para a comunicação entre sistemas CAD, através da troca de arquivos gráficos.

No caso do Editor Gráfico, optou-se por um padrão próprio para o arquivo gráfico, justamente para que o mesmo seja o mais simples possível, tornando-o facilmente manipulável. Se for o caso, pode-se criar as interfaces de conversão para o padrão IGES, e o Editor Gráfico poderia se comunicar com os diversos sistemas já existentes.

4.5. Montagem do arquivo

O arquivo é composto de duas linhas iniciais, e os parâmetros definidores dos elementos a seguir. A primeira linha contém o nome do arquivo. Esta linha não tem importância para o Editor e, portanto, qualquer comentário sobre o desenho pode ser ali colocado. A segunda linha do arquivo contém três valores. O primeiro é o fator de escala aplicado ao desenho para definir a janela de visão que estava ativa durante a última operação de gravação. Os outros dois números completam esta informação, indicando o ponto de centragem do desenho na janela de visão, em coordenadas de desenho.

As informações que estão a seguir dentro do arquivo são os parâmetros que definem todos os elementos que compõe o desenho. Considerando agora cada definição de elemento, pode-se dividi-la em três partes. A primeira parte, ou primeiro número é o cabeçalho da definição do elemento. Neste número estão registra-

NNNN - número de ordem do elemento no arquivo.

E - código do elemento.

FF - folha (nível) de desenho ao qual o elemento pertence.

A - tipo de linha do elemento.

Figura 4.1 - O cabeçalho de cada elemento.

dos o número de ordem do elemento dentro do arquivo, o código do elemento, indicando de que elemento se trata (linha, arco, círculo, etc), o atributo de linha deste elemento, identificando que tipo de linha é utilizado para o desenho do mesmo, e finalmente a folha (ou nível) de desenho á qual ele pertence. Na figura 4.1 pode-se ver com mais clareza a montagem da linha de cabeçalho.

Cada elemento ou entidade geométrica possui um código de identificação, que consta no cabeçalho da definição do elemento. Este código pode ser melhor visualizado na tabela 4.1.

CODIGO	ENTIDADE GEOMETRICA
1	linha
2	linha poligonal
3	retângulo
4	polígono
5	círculo
8	arco
9	texto

Tabela 4.1 - Código dos elementos.

A segunda parte da definição de um elemento no arquivo contém quatro coordenadas. Estas coordenadas especificam os cantos inferior esquerdo e superior direito de um retângulo de lados horizontais e verticais. Este é o menor retângulo que envolve o elemento em questão. O retângulo envolvente é o método utilizado pelo programa para a identificação do elemento na tela. Para o usuário não é difícil imaginar, para um determinado elemento na tela, o seu retângulo envolvente. Portanto, para sua identificação basta posicionar o cursor dentro deste retângulo imaginário e confirmar a escolha. A lista de elementos na memória é pesquisada e vão sendo mostrados, um a um, os elementos onde o cursor está no interior de seu retângulo envolvente, até que o usuário escolha por um deles ou cancele a busca.

Finalmente, a última parte da definição de um elemento é a lista de parâmetros que informam posição, forma e tamanho do elemento. São as coordenadas dos pontos chave que formam o elemento, raios, ângulos, etc.

A linha é definida por apenas dois pares de coordenadas indicando seus pontos extremos. O retângulo também é definido por apenas dois pares de coordenadas, que são seus extremos inferior esquerdo e superior direito. O retângulo é apenas uma simplificação do polígono, que será visto a seguir. Seus lados são horizontais e verticais e possui quatro pontos identificadores, para que o mesmo possa sofrer rotações. Para sua definição, na entrada de dados, bastam dois pontos indicando sua diagonal. O círculo é definido simplesmente pelas coordenadas de seu centro e pelo seu raio. O arco, além destes dados, tem com parâmetros o ângulo inicial e o ângulo final, contados a partir da horizontal e no sentido anti-horário.

A linha poligonal e o polígono são definidos da mesma maneira. A diferença está no desenho do elemento, já que o polígono é uma linha poligonal fechada. Como o número de pontos de uma linha poligonal ou polígono pode ser qualquer, esta informação é necessária no arquivo. Desta maneira, para os dois elementos em questão, o primeiro parâmetro desta parte da definição indica o número de pares de coordenadas que vem a seguir.

O texto é um elemento que, além do texto propriamente dito, armazenado em um vetor de caracteres ou *string*, possui um par de coordenadas que indica o seu ponto inicial, o ângulo do texto, que pode ser de 0 a 360°, e o tamanho dos caracteres, definido em função da altura dos mesmos, uma vez que na largura é mantida sempre a mesma proporção. Na linha do texto, os quatro

primeiros caracteres são "TXT " para informar ao Editor que se trata de uma *string*. Na tabela 4.2 pode-se observar a forma do

AFF1NNNN	AFF2NNNN	AFF3NNNN	AFF4NNNN
xmin	xmin	xmin	xmin
ymin	ymin	ymin	ymin
xmax	xmax	xmax	xmax
ymax	ymax	ymax	ymax
X1	num-par	X1	num-par
Y1	X1	Y1	X1
X2	Y1	-	Y1
Y2	-	-	-
	Xn	X4	Xn
	Yn	Y4	Yn

AFF5NNNN	AFF8NNNN	AFF9NNNN
xmin	xmin	xmin
ymin	ymin	ymin
xmax	xmax	xmax
ymax	ymax	ymax
Xc	Xc	texto
Yc	Yc	X
raio	raio	Y
	θ	ângulo
	θ_f^i	altura

Tabela 4.2 - A especificação do arquivo gráfico.

arquivo gráfico para cada elemento, contendo seu cabeçalho, o retângulo envolvente e as coordenadas dos pontos.

Todas as coordenadas e grandezas até agora referidas estão armazenadas no arquivo gráfico em unidades reais. A partir disto fica mais fácil para o projetista a manipulação dos dados e a aplicação do arquivo gráfico para entrada de dados de um

programa qualquer de dimensionamento, bem como para a impressão do desenho, bastando escolher uma escala adequada para o tamanho da folha na qual se deseja o desenho.

CAPITULO 5

PROJETO DE MOLAS

5.1. Introdução

O projeto de molas é geralmente um trabalho que requer bastante tempo do projetista, em virtude do grande número de parâmetros envolvidos. Na busca do projeto que melhor atende às especificações, deve-se arbitrar um conjunto de parâmetros e calcular outros. O resultado deve ser verificado e, caso não seja o desejado, novo conjunto de parâmetros deve ser arbitrado.

Quando se deseja otimizar o projeto com relação a um determinado parâmetro [46,47,48,49], como diâmetro externo mínimo, por exemplo, o número de tentativas aumenta bastante, pois a faixa de resultados aceitos é mais restrita. Pode-se então deduzir expressões para, a partir de um conjunto de parâmetros arbitrados, chegar-se sempre à mola com o valor mínimo com relação ao critério de projeto escolhido. Estas expressões podem então ser codificadas em um programa computacional, de forma que o usuário possa arbitrar um conjunto inicial de parâmetros e, rapidamente, ter o resultado para a verificação. A

cações foram satisfeitas ou é necessário novo cálculo com outro conjunto de parâmetros.

Este trabalho iterativo tem maior eficiência quando o desenho da mola calculada é visualizado pelo projetista. A partir dos resultados obtidos no cálculo da mola, pode-se converter as dimensões calculadas em coordenadas, e construir um arquivo gráfico. Este arquivo é então interpretado pelo Editor Gráfico e o desenho da mola mostrado no vídeo. O projetista tem condições de realizar alterações no desenho (adição de detalhes ou comentários, legenda, etc) por meio dos comandos do Editor, tirar uma cópia em papel numa impressora ou traçador gráfico, ou ainda simplesmente voltar ao programa e recalcular a mola com um novo conjunto de parâmetros.

5.2. Formulário básico

Este programa limita-se ás molas helicoidais de compressão. Admite-se também somente molas que apresentam uma relação força/deflexão linear, ou seja,

$$\frac{F}{d} = \text{constante} = K \qquad (5.1)$$

onde

F : carga axial que atua na mola;

d : deflexão da mola;

K : constante elástica ou constante de rigidez da mola;

A tensão (T) que atua na mola é dada por

$$T = K_w \frac{8FD_m}{\pi D_a^3} = K_w \frac{8FC}{\pi D_a^2} \qquad (5.2)$$

onde

T : tensão de cisalhamento do material da mola;

D_m : diâmetro médio da mola;

D_a : diâmetro do fio (arame);

C : índice de curvatura = D_m/D_a;

K_w : fator de Wahl,

$$K_w = \frac{4C - 1}{4C - 4} + \frac{0,615}{C} \qquad (5.3)$$

A deflexão da mola (d) pode ser calculada pela expressão

$$d = \frac{\theta D_m}{2} = \frac{64FR_m^3N_a}{D_a^4G} = \frac{8FD_m^3N_a}{D_a^4G} \qquad (5.4)$$

onde

θ : ângulo de deformação por torção;

R_m : raio médio = $D_m/2$;

N_a : número de espiras ativas;

G : módulo de elasticidade transversal;

A extremidade da mola pode ter diversos tipos de acabamento. O tipo de extremidade tem influência sobre o número de espiras ativas da mola,e sobre seu comprimento. O número de espiras ativas (N_a) pode ser obtido a partir da expressão (5.4), ou seja,

$$N_a = \frac{D_aGd}{8FC_3} \qquad (5.5)$$

TIPO DE EXTREMIDADE	ESPIRAS		COMPRIMENTO DA MOLA	
	totais (N)	inativas (N_i)	livre (L)	sólido (H)
EM PONTA	$N_a + 1$	1	$pN_a + D$	$D(N_a + 1)$
EM ESQUADRO	$N_a + 2$	2	$pN_a + 3D$	$D(N_a + 3)$
EM ESQUADRO ESMERILHADA	$N_a + 2$	2	$pN_a + 2D$	$D(N_a + 2)$

Tabela 5.1 - Influência dos tipos de extremidades segundo [46].

O número total de espiras será então a soma das espiras ativas e inativas, ou seja,

$$N = N_a + N_i \qquad (5.6)$$

A influência do tipo de extremidade sobre o número de espiras ativas e o comprimento da mola [46], está mostrado na tabela 5.1.

5.3. Critérios de projeto

A partir do formulário básico pode-se então deduzir expressões que permitem a determinação do valor exato para o índice de curvatura da mola (C), seguindo algum critério para minimizar o valor de determinado parâmetro. O índice de curvatura é obtido em função de uma constante de projeto (B), que é característica para cada um dos critérios de cálculo

da mola. Neste programa estão previstos os seguintes critérios:

- mola com peso mínimo.
- mola com comprimento mínimo.
- mola com diâmetro externo especificado.
- mola com diâmetro interno especificado.

Devido ao fato das expressões encontradas para cálculo do índice de curvatura da mola em função da constante de projeto serem muito extensas, é comum utilizar a representação gráfica destas expressões, como mostram as curvas da figura 5.1. No programa foram utilizadas as expressões, de modo que o resultado é, além de mais rápido, mais preciso que os valores encontrados pelo procedimento usual de cálculo. O formulário para cada um dos critérios de projeto acima citados está a seguir.

5.3.1. Mola com peso mínimo

O peso da mola (P) é dado por

$$P = \rho(N_a + N_i)(\pi D_m)\frac{\pi D_a^2}{4} \qquad (5.6)$$

de onde

$$\frac{4P}{\rho\pi^2} = N_a C D_a^3 + N_i C D_a^3 \qquad (5.7)$$

Substituindo na equação (5.7) os valores de D_a e N_a obtidos das equações (5.2) e (5.4) respectivamente, deixando no segundo membro somente os termos em K_w e C, e derivando-se em relação a C e fazendo dP/dC = 0 encontra-se uma igualdade com

termos de K_w e C no segundo membro e a constante de projeto (B), definida por

$$B = \frac{Gd}{N_i(8F\pi T)^{1/2}} \qquad (5.8)$$

no primeiro membro. Substituindo os termos em K_w por seu valor da expressão (5.3), e de sua derivada, obtem-se a curva A da figura 5.1, dada por

$$B = \frac{C^3(C^3-0,635C^2-0,980C+0,615)^{1/2}(5C^3-7,27C^2-1,21C+1,23)}{4(1,365C^4-0,732C^3-0,673C^2+0,981C-0,378)} \qquad (5.9)$$

5.3.2. *Mola com comprimento mínimo*

O comprimento livre da mola é dado por

$$L = (N_a + N_i)D_a + \epsilon \qquad (5.10)$$

onde $\epsilon = d$ + folga, sendo a folga igual a diferença entre a deflexão máxima (mola fechada) e a deflexão com carga máxima de trabalho.

Seguindo o mesmo procedimento do critério do peso mínimo, encontramos a forma da equação a seguir, representada pela curva B na figura 5.1.

$$B = \frac{C^4}{2}\left[\frac{C-1}{C^2+0,365C-0,615}\right]^{1/2} \times \left[\frac{C^2-2C+0,25}{2C^3+0,095C^2-3,19C+1,845}\right] \qquad (5.11)$$

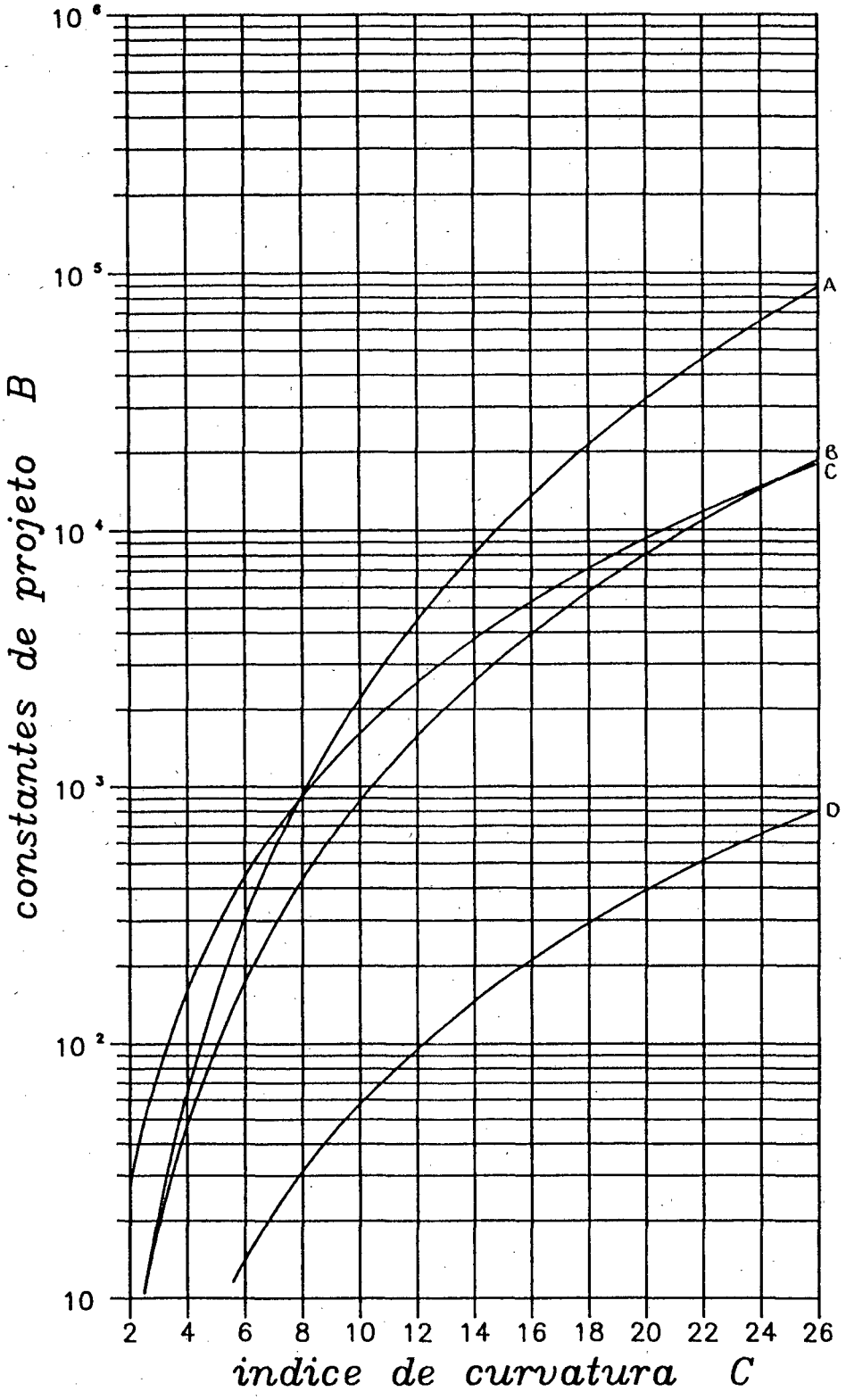


Figura 5.1 - As curvas para seleção do índice de curvatura.

5.3.3. Mola com diâmetro externo especificado

Quando, além dos parâmetros usuais de entrada (F, d, G, e T), se especifica um diâmetro, uma única solução é encontrada para o problema. Neste caso, o índice de curvatura pode ser encontrado da seguinte maneira

$$D_e = D_m + D_a = D_a(C + 1) = \left[\frac{8FK_w C}{\pi T}\right]^{1/2}(C + 1) \qquad (5.12)$$

Substituindo o fator de Wahl, da expressão (5.3), encontramos

$$D_e^2\left[\frac{\pi T}{8F}\right] = \frac{C^4 + 2,365C^3 + 1,115C^2 - 0,867C - 0,615}{C - 1} \qquad (5.13)$$

onde o primeiro membro é a constante de projeto (B), ou seja,

$$B = D_e^2\left[\frac{\pi T}{8F}\right] \qquad (5.14)$$

A curva C da figura 5.1 representa a expressão (5.13), considerando a constante de projeto acima definida.

5.3.4. Mola com diâmetro interno especificado

Da mesma maneira que no caso anterior, o índice de curvatura (C) será calculado por

$$D_i = D_m - D_a = D_a(C - 1) = \left[\frac{8FK_w C}{\pi T}\right]^{1/2}(C - 1) \qquad (5.15)$$

Substituindo o fator de Wahl, da expressão (5.3), encontramos

$$D_i^2\left[\frac{\pi T}{8F}\right] = C^3 - 0,635C^2 - 0,98C + 0,615 \qquad (5.16)$$

onde o primeiro membro é a constante de projeto (B), ou seja,

$$B = D_i^2\left[\frac{\pi T}{8F}\right] \qquad (5.17)$$

e a curva representativa da equação (5.16) é a curva D do diagrama da figura 5.1.

5.4. A interface para o Editor

A partir dos dados gerados pelo programa de cálculo de molas, pode-se gerar o arquivo gráfico contendo os elementos geométricos básicos que formam o desenho da mola calculada. Para a criação do arquivo, é necessária a escolha da região no espaço de desenho do Editor onde será criado o desenho da mola. Isto é equivalente a escolher, numa folha de papel, onde será começado o desenho.

Para tornar mais clara e simples a geração de um arquivo gráfico a partir dos dados calculados de uma mola, tomou-se o exemplo da mola desenhada na figura 5.2. Por questão de facilidade, posicionou-se o extremo inferior esquerdo do desenho da mola na origem do sistema de coordenadas. O primeiro elemento a ser gerado pode ser, então, o círculo inferior da primeira espira da mola. Para a definição de um círculo segundo a especificação do arquivo gráfico do Editor (veja capítulo 4 - O Arquivo

Gráfico), deve-se ter as coordenadas de seu centro e o raio. Neste caso, o centro do círculo terá as coordenadas (R,R), onde R é o raio do arame da mola, e o raio do círculo será obviamente R.

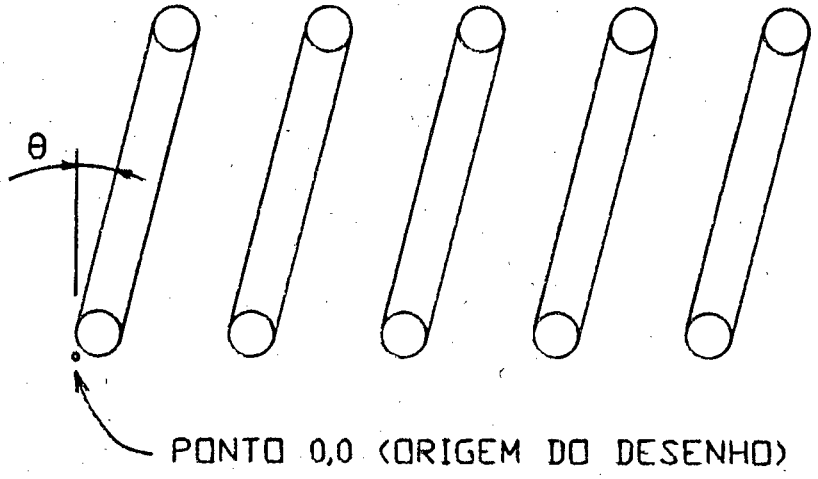


Figura 5.2 - O desenho de uma mola.

O outro círculo da mesma espira tem as coordenadas do círculo anterior acrescidas de passo/2 na direção do eixo X e do diâmetro médio (d_m) na direção Y. Desta maneira temos seu centro em (R+passo/2 , R+d_m) e o raio igual a R. Conforme a especificação do arquivo gráfico, temos a seguinte sequência de dados para a definição dos dois círculos

150001
0
0
2R
2R
R
R
R
150002
passo/2
d_m
passo/2+2R
d_m+2R
passo/2+R
d_m+R
R

Para completar o desenho desta espira falta incluir no arquivo as duas linhas. Estas linhas tem suas coordenadas definidas segundo os pontos de tangência aos círculos anteriores. Assim, conforme a figura 5.2, temos, para a linha da esquerda seu primeiro ponto com coordenadas (R(1-cosθ) , R(1+senθ)). Para obter-se o segundo ponto, basta somar ãs coordenadas do primeiro ponto passo/2 na direção X e d_m na direção Y, resultando em (R(1-cosθ) + passo/2 , R(1+senθ) + d_m). A outra linha pode ser obtida do mesmo modo, sendo seus pontos (R(1+cosθ) , R(1-senθ)) e (R(1+cosθ) + passo/2 , R(1-senθ) + d_m). Os dados no arquivo referentes as duas linhas terão o seguinte formato:

110003
R-cosθ
R+senθ
R-cosθ+passo/2
R+senθ+d_m
R-cosθ
R+senθ
R-cosθ+passo/2
R+senθ+d_m
110004
R+cosθ
R-senθ
R+cosθ+passo/2
R-senθ+d_m
R+cosθ
R-senθ
R+cosθ+passo/2
R-senθ+d_m

Tem-se assim uma espira completa montada no arquivo gráfico. Pode-se criar um procedimento composto das etapas anteriores para a geração de uma espira, e criar a mola toda, repetindo-se este procedimento com um incremento na direção X no valor de um passo da mola, para cada espira desenhada. O número de espiras é um dado calculado pelo programa.

Para que o desenho da mola apareça na tela totalmente enquadrado, deve-se calcular a escala e o ponto de centro do desenho. Estes dados devem constar na segunda linha do arquivo gráfico, conforme a especificação para o Editor. Os outros elementos, como linhas de cota ou de simetria, cotas, etc, podem ser criados no arquivo seguindo-se procedimento similar ao descrito para as espiras.

5.5. Um projeto de mola

A seguir estão os resultados numéricos e gráficos obtidos pelo programa de dimensionamento de molas. Com um conjunto de dados de entrada, foi rodado o programa para os quatro critérios de projeto disponíveis e também para os três tipos de extremidades.

Supondo uma mola a ser dimensionada, utilizando um aço qualidade corda de piano (ASTM A-228), com módulo torcional de 85000 MPa e tensão admissível 900 MPa. Deseja-se uma mola para uma carga de 1000 N, provocando uma deflexão de 50 mm. A mola deve ser dimensionada para

A - peso mínimo.

B - diâmetro externo de 50 mm.

C - diâmetro interno de 35 mm.

5.5.1. Mola com peso mínimo

A sequência de telas a seguir mostra a entrada de dados e os resultados numérico e gráfico do cálculo da mola para o peso mínimo.

```
EGG.PM   v2.01     (03.06.88)      LABORATORIO CAE/CAD . GRANTE - UFSC

   PROMOL  v1.00   PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Escolha o parametro base para o projeto da mola : 1

     1.  Peso minimo
     2.  Diametro externo
     3.  Diametro interno
     4.  Comprimento minimo

Escolha o tipo de extremidade da mola : 2

     1.  Em esquadro esmerilhada
     2.  Em esquadro
     3.  Em ponta


Calcula mola com extremidade em esquadro
        com base no seu peso minimo.  Confirma ? (S/N)  [S]
```

Figura 5.3A - Escolha do critério de projeto.

```
EGG.PM    v2.01      (03.06.88)        LABORATORIO  CAE/CAD    GRANTE - UFSC

    PROMOL   v1.00     PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Entre com os dados de projeto :

     carregamento (N)          = 1000
     deflexao (mm)             = 50
     tensao admissivel (MPa)   = 900
     modulo torcional (MPa)    = 85000


Os dados estao corretos ? (S/N)   [S]
```

Figura 5.3B - Entrada de dados.

```
EGG.PM    v2.01      (03.06.88)        LABORATORIO  CAE/CAD    GRANTE - UFSC

    PROMOL   v1.00     PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Dados da mola calculada :

     Diametro do arame         = 4.6946 mm
     Diametro medio da mola    = 29.4446 mm
     Diametro interno          = 24.7500 mm
     Diametro externo          = 34.1392 mm
     Numero total de espiras   = 12.1 (2 inativas)
     Comprimento livre         = 111.5882 mm
     Comprimento solido        = 56.8428 mm
     Passo da mola             = 11.0394 mm

Selecione opcao :

     1. Recalcular a mola
     2. Calcular outra mola
     3. Gerar o arquivo grafico para a mola calculada
```

Figura 5.3C - Resultado numérico.

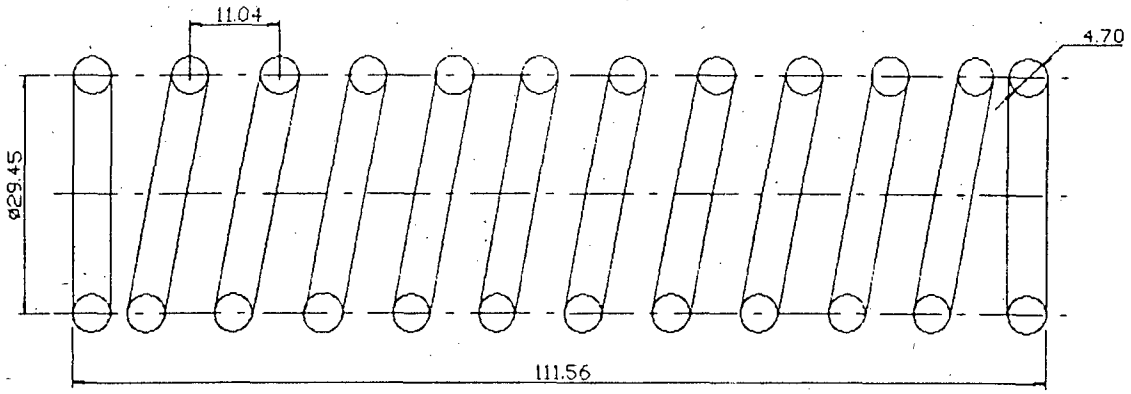


Figura 5.3D - Resultado gráfico, a mola no Editor.

5.5.2. Mola com diâmetro externo especificado

```
EGG.PM    v2.01      (03.06.88)        LABORATORIO CAE/CAD    GRANTE - UFSC

    PROMOL  v1.00     PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Escolha o parametro base para o projeto da mola : 2

     1.  Peso minimo
     2.  Diametro externo
     3.  Diametro interno
     4.  Comprimento minimo

Escolha o tipo de extremidade da mola : 3

     1.  Em esquadro esmerilhada
     2.  Em esquadro
     3.  Em ponta


Calcula mola com extremidade em em ponta
          com base no seu diametro externo esp.  Confirma ? (S/N)  [S]
```

Figura 5.4A - Escolha do critério de projeto.

```
EGG.PM   v2.01     (03.06.88)       LABORATORIO  CAE/CAD    GRANTE - UFSC

   PROMOL   v1.00    PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Entre com os dados de projeto :

     carregamento (N)          = 1000

     deflexao (mm)             = 50

     tensao admissivel (MPa)   = 900

     modulo torcional (MPa)    = 85000

     diametro externo (mm)     = 50

Os dados estao corretos ? (S/N)   [S]
```

Figura 5.4B - Entrada de dados.

```
EGG.PM   v2.01     (03.06.88)       LABORATORIO  CAE/CAD    GRANTE - UFSC

   PROMOL   v1.00    PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Dados da mola calculada :

     Diametro do arame         = 5.2976 mm
     Diametro medio da mola    = 44.7283 mm
     Diametro interno          = 39.4307 mm
     Diametro externo          = 50.0260 mm
     Numero total de espiras   = 5.7 (1 inativas)
     Comprimento livre         = 82.5462 mm
     Comprimento solido        = 30.0690 mm
     Passo da mola             = 17.6533 mm

Selecione opcao :

     1. Recalcular a mola
     2. Calcular outra mola
     3. Gerar o arquivo grafico para a mola calculada
```

Figura 5.4C - Resultado numérico.

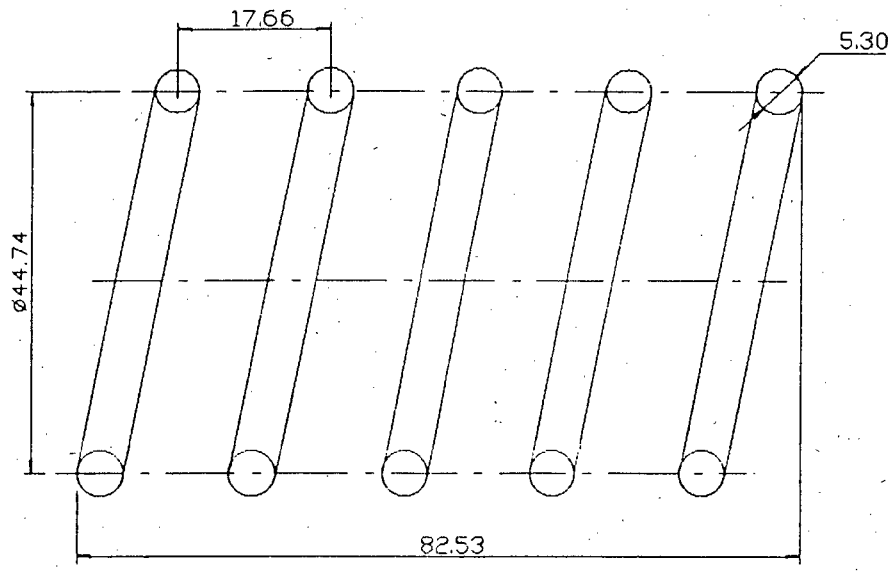


Figura 5.4D - Resultado gráfico, a mola no Editor.

5.5.3. Mola com diâmetro interno especificado

```
EGG.PM   v2.01      (03.06.88)        LABORATORIO CAE/CAD   GRANTE - UFSC

   PROMOL   v1.00    PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Escolha o parametro base para o projeto da mola : 3

     1.  Peso minimo
     2.  Diametro externo
     3.  Diametro interno
     4.  Comprimento minimo

Escolha o tipo de extremidade da mola : 1

     1.  Em esquadro esmerilhada
     2.  Em esquadro
     3.  Em ponta


Calcula mola com extremidade em esquadro esmerilhada
         com base no seu diametro interno esp. Confirma ? (S/N)  [S]
```

Figura 5.5A - Escolha do critério de projeto.

```
EGG.PM    v2.01     (03.06.88)      LABORATORIO  CAE/CAD    GRANTE - UFSC

    PROMOL   v1.00    PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Entre com os dados de projeto :

     carregamento (N)          = 1000

     deflexao (mm)             = 50

     tensao admissivel (MPa)   = 900

     modulo torcional (MPa)    = 85000

     diametro interno (mm)     = 35

Os dados estao corretos ? (S/N)   [S]
```

Figura 5.5B - Entrada de dados.

```
EGG.PM    v2.01     (03.06.88)      LABORATORIO  CAE/CAD    GRANTE - UFSC

    PROMOL   v1.00    PROJETO DE MOLAS HELICOIDAIS DE COMPRESSAO

Dados da mola calculada :

     Diametro do arame         = 5.1320 mm
     Diametro medio da mola    = 40.1509 mm
     Diametro interno          = 35.0189 mm
     Diametro externo          = 45.2829 mm
     Numero total de espiras   = 7.7 (2 inativas)
     Comprimento livre         = 92.4044 mm
     Comprimento solido        = 39.4825 mm
     Passo da mola             = 16.2302 mm

Selecione opcao :

     1. Recalcular a mola
     2. Calcular outra mola
     3. Gerar o arquivo grafico para a mola calculada
```

Figura 5.5C - Resultado numérico.

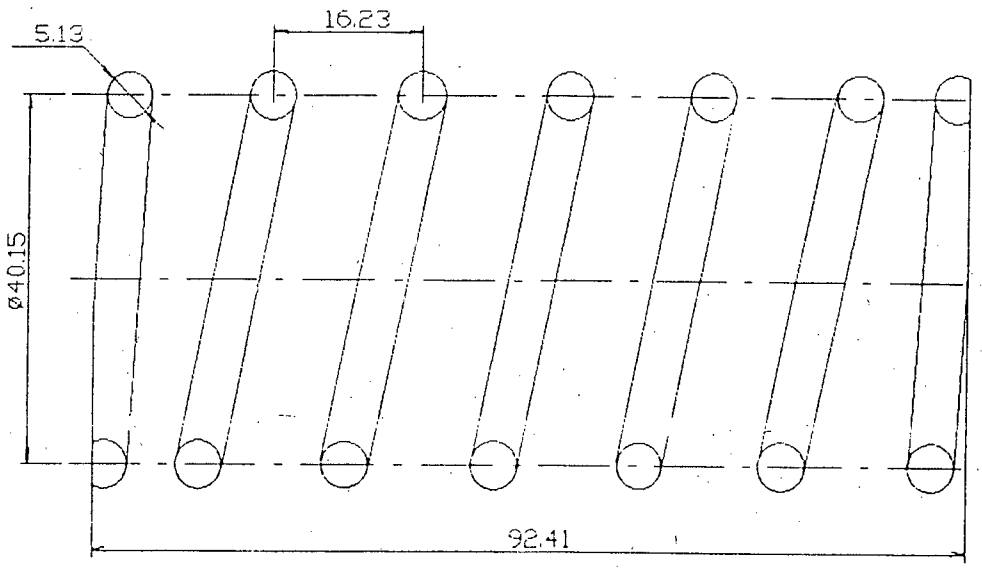


Figura 5.5D - Resultado gráfico, a mola no Editor.

CAPITULO 6

CONCLUSOES

6.1. Conclusões

Como pode ser observado no exemplo de aplicação mostrado no Capítulo 5 - Projeto de Molas, o Editor Gráfico cumpriu seu papel fornecendo uma saída gráfica bastante poderosa para um programa aplicativo de Engenharia Mecânica, comunicando-se com o mesmo através do arquivo gráfico.

No caso, o programa para cálculo de molas helicoidais de compressão gera, a partir dos dados da mola calculada (dimensões), um arquivo no formato especificado pelo Editor, composto de elementos gráficos e suas coordenadas. Este arquivo é então interpretado pelo Editor e o desenho da mola é mostrado na tela de vídeo. A partir daí, os recursos do Editor podem ser utilizados para alterar o desenho da mola, ou mesmo acrescentar ítens como legenda, informações adicionais, listas de materiais, etc, ou então tirar uma impressão da mola num traçador gráfico ou numa impressora matricial.

A construção da interface de conversão das dimensões da mola calculada em coordenadas e elementos para o arquivo gráfico

do Editor é bastante simples, envolvendo apenas algumas transformações geométricas. O que se mostrou um inconveniente é a necessidade de especificação no arquivo gráfico do retângulo envolvente de cada elemento. Além de aumentar o tempo de processamento e o número de operações na interface, estes dados provocam um aumento considerável no arquivo gráfico.

Quanto às saídas gráficas em papel, observou-se que a impressora matricial é uma opção de baixo custo, mas que oferece uma saída de qualidades limitadas. Tanto a resolução como o limite de tamanho do papel são fatores negativos que pesam muito. Entretanto, dependendo do tipo de aplicação, pode-se utilizar uma impressora com resultados satisfatórios, como é o caso de layouts ou desenhos pequenos (A4) sem muitos detalhes. Para aplicações mais específicas de projeto mecânico, um traçador gráfico (plotter) é indispensável. O resultado final de uma saída em plotter é muito bom, e pode-se escolher cor, tipo de pena, tipo e tamanho de papel, etc.

6.2. Recomendações

Para melhorar o desempenho do sistema, são necessárias modificações no programa, incluindo algoritmos mais rápidos para as funções gráficas básicas, como os geradores de pontos, linhas, círculos e arcos. Deste modo operações como a atualização tornam-se mais rápidas, e o trabalho com o Editor fica mais cômodo para o projetista.

Deve-se estudar uma maneira para eliminar os dados relativos ao retângulo envolvente de cada elemento do arquivo gráfico. Estas informações estão tornando o arquivo gráfico muito grande, e dificultando a construção de interfaces para os programas aplicativos. Para eliminar o retângulo envolvente deve-se elaborar outro algoritmo para a identificação de um elemento no desenho por posicionamento do cursor, ou colocar as informações do retângulo envolvente somente no arquivo em memória, não passando o mesmo para o disco. Assim seria economizado espaço em disco e a montagem do arquivo a partir de programas aplicativos seria simplificada.

Foram desenvolvidos protocolos de comunicação para impressoras matriciais com padrão de impressão Epson, o que satisfaz as principais impressoras do mercado nacional, como a Grafix, Emília PC, Mônica, Diana e Rima. Seria interessante uma opção de configuração mais elaborada, indicando se a impressora é 132 ou 80 colunas, e utilizando o máximo dos recursos de cada uma.

No caso do traçador gráfico, foi desenvolvido o *driver* para o TDD 21R, traçador de uma só pena da Digicon, que utiliza a linguagem de programação da Houston Instruments, e para o traçador Hewllet Packard modelo HP7595, que utiliza a HPGL, linguagem de programação da HP. Seria interessante fornecer também *drivers* para o modelo TDD 43, da Digicon, que é o modelo menor e utiliza a linguagem de programacão também da Hewlett

Packard (HPGL), para os outros modelos da HP, e também para os modelos da Smar encontrados no mercado nacional.

O Editor apresenta apenas os recursos básicos para criação e manipulação de entidades gráficas. É interessante que sejam adicionadas funções auxiliares para desenho, de forma a dar ao projetista melhores condições para elaborar um desenho. Funções como arcos de concordância e chanfros, linhas paralelas ou perpendiculares a outros elementos, cotagem automática, etc, devem ser implementadas para melhorar os recursos de desenho do Editor.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] *Turbo Pascal Reference Manual v3.0*, Borland, 1985.

[2] *Turbo Pascal Owner's Handbook v4.0*, Borland, 1987.

[3] *Graphix Toolbox Owner's Handbook v1.0*, Borland, 1985.

[4] Wood, S., *Using Turbo Pascal*, McGraw-Hill, 1986.

[5] Farrer, H. et al., *Pascal Estruturado*, Guanabara Dois, 1985.

[6] Rugg, Tom and Feldman, Phil, *Turbo Pascal Program Library*, Que, 1986.

[7] Jamsa, K. and Nameroff, S., *Turbo Pascal Programmer Library*, McGraw-Hill, 1986.

[8] Schildt, Herbert, *Advanced Turbo Pascal*, McGraw-Hill, 1986.

[9] Wood, Steve, *Using Turbo Pascal*, McGraw-Hill, 1986.

[10] Mazlack, L.J., *Structured Problem Solving with Pascal*, CBS College Publishing, 1983

[11] *Disk Operating System Technical Reference*, IBM, 1985.

[12] Norton, Peter, *Inside the IBM PC*, Brady, 1986.

[13] Norton, Peter, *Programmer's Guide to the IBM PC*, Microsoft Press, 1985.

[14] Wadlow, T.A., *Memory Resident Programming on the IBM PC*, Addison-Wesley, 1987.

[15] Magalhães, L.P., *Computação Gráfica*, Papirus, 1986.

[16] Cunha, G.J. et al., *Computação Gráfica e suas aplicações em CAD*, Atlas, 1987.

[17] Persiano, R.C.M. e Oliveira, A.A.F., *Introdução à Computação Gráfica*, LTC Editora, 1988.

[18] Berger, Mark, *Computer Graphics with Pascal*, Benjamin Cummings, 1986.

[19] Newman, W.M. and Sproull, R.F., *Principles of Interactive Computer Graphics*, McGraw-Hill, 1982.

[20] Foley, J.D. and Van Dan, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

[21] Santo, H.P., *Métodos Gráficos e Geometria Computacionais*, Dinalivro, 1984.

[22] Hyman, M.I., *Advanced IBM PC Graphics - State of the Art*, Brady, 1985.

[23] Harrington, S., *Computer Graphics - A Programming Aproach*, McGraw-Hill, 1983.

[24] Cunha, G.J. et al., *Computação Gráfica - O Padrão GKS*, Atlas, 1987.

[25] Hopgood, F.R.A. et al., *Introduction to the Graphical Kernel System (GKS)*, Academic Press, 1983.

[26] *Initial Graphics Exchange Specification (IGES)*, U.S. Department of Commerce, 1983.

[27] Pao, Y.C., *Elements of Computer-Aided Design and Manufacturing*, John Wiley and Sons, 1984.

[28] Besant, C.B., *CAD/CAM - Projeto e Fabricação com Auxílio do Computador*, Campus, 1985.

[29] Tozzi, C.L., *PAC - Projeto Auxiliado por Computador*, Papirus, 1986.

[30] Voisinet, D.D., *CADD - Projeto e Desenho Auxiliados por Computador*, McGraw-Hill, 1988.

[31] Hatfield, L. and Herzog, B., *Graphics Software - From Techniques to Principles*, Computer Graphics and Applications, JAN 1982, pp 59 -80.

[32] Siebers, G.R., *An Introduction to Computer Graphics*, Computer-Aided Design, ABR 1986, vol 18 n3 pp 161-179.

[33] Marshall, P.S., *Do It Yourself CAD/CAM*, Machine Design, MAI 1982, vol 54 n10, pp 74-79.

[34] Kilgour, A.C., *A Hierarchical Model of a Graphics System*, Computer Graphics, ABR 1981, vol 15 n1, pp 35-47.

[35] Krouse, J.K., *Software for Mechanical Design*, Machine Design, OUT 1982, vol 54 n23, pp 42-48.

[36] Krouse, J.K., *Off the Shelf Software for Mechanical Design*, Machine Design, ABR 1983, vol 55 n9, pp 47-52.

[37] Constantinou, S. et al., *Evaluation Procedures to be Used During the Development of CAD Systems*, Computer Aided Design, NOV 1982, vol 14 n6, pp 321-328.

[38] Yamada, Jiro, *A Low Cost Drafting System Based on a PC*, Computer Graphics and Applications, MAI 1984, pp 61-65.

[39] Clements, R.R., *A Three Layer Philosophy for the Design of CAD Software*, Computer Aided Design, AGO 1985, vol 17 n6, pp 262-265.

[40] *AutoCad User Guide*, AutoDesk Inc., 1985.

[41] Raker, D. and Rice, H., *Inside AutoCad*, New Riders, 1986.

[42] *Cadtec - Manual do Usuário*, Itautec, 1986.

[43] *Prodesign II User's Guide*, American Small Business Computers, 1986.

[44] *VersaCad Advanced User's Manual*, T&W Systems, 1985.

[45] Shigley, J.E. and Mitchell, L.D., *Mechanical Engineering Design*, McGraw-Hill, 1983.

[46] Wahl, A.M., *Mechanical Springs*, Penton Publishing Company, 1944.

[47] Carvalho, J.R. e Moraes, P., *Órgãos de Máquinas - Dimensionamento*, LTC Editora, 1978.

[48] Chironis, N.P., *Spring Design and Application*, McGraw-Hill, 1961.

[49] Dietrich, Al, *Home Computers Aid Spring Design*, Design Engineering, JUN 1981, pp 31-36.

[50] Hall, A.S. and Holowenko, A.R., *Elementos Orgânicos de Máquinas*, McGraw-Hill, 1977.

[51] *Traçador Gráfico TDD-21R - Manual de Programação*, Digicon, 1985.

[52] Mirshawka, V., *Imprimindo Maravilhas com a Grafix*, Nobel, 1985.

APENDICE 1

ALGORITMOS DE PROGRAMAÇAO

A1.1. Mapeamento *Window-Viewport*

Atualmente encontra-se no mercado um grande número de periféricos e dispositivos gráficos, como impressoras matriciais, traçadores gráficos e vídeos. Estes dispositivos diferem entre si não só quanto ao modo de operação ou princípio de funcionamento. Utilizam também diferentes unidades de medida (milímetro, polegada, etc), diferentes tamanhos da área útil para desenho (tamanho da tela no caso de vídeos e espaço útil no papel para o caso de impressoras matriciais e traçadores gráficos), bem como uma resolução gráfica (número de pontos representados por unidade de medida) característica de cada dispositivo. Esta resolução é comumente diferente nas direções X e Y dos eixos coordenados, fato pelo qual foi definido o *fator de aspecto*, que é a relação entre a resolução horizontal e a resolução vertical. Desta maneira, para se obter a correta representação de um objeto em um determinado dispositivo, necessita-se de uma transformação de coordenadas particular.

No sentido de padronizar o processo de mapeamento de coordenadas, foram introduzidos os conceitos de *window* e

viewport [17,18,19,20,23]. Ambos são retângulos de lados paralelos aos eixos coordenados, sendo que a *window* é definida no sistema de coordenadas do usuário, ou seja, são as coordenadas que devem sofrer a transformação. A *viewport* é definida no sistema de coordenadas do dispositivo e é, portanto, o resultado da transformação.

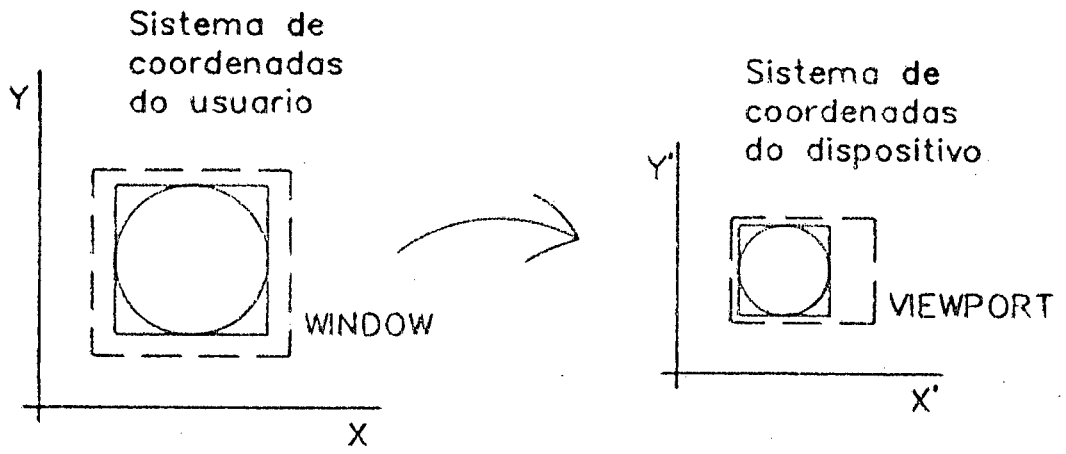


Figura A1.1 - Mapeamento *Window-Viewport*.

No caso do Editor Gráfico, o sistema de coordenadas do usuário é cartesiano, sendo utilizada a unidade de desenho como medida. Quando o desenho é impresso em um dispositivo de saída, a unidade de desenho é associada a alguma unidade de medida, como o milímetro ou polegada, e o desenho é devidamente escalado para permitir seu correto enquadramento no papel.

Quando se deseja mapear um objeto definido no sistema de coordenadas do usuário no sistema de coordenadas de um dispositivo, de modo que o objeto seja completamente enquadrado na área de desenho do dispositivo, tem-se duas situações comuns. Pode-se desenhar o objeto centrado em relação a área de desenho, ou

então posicioná-lo segundo a origem do dispositivo, geralmente o canto inferior esquerdo da área útil de desenho. Na figura A1.2 pode-se ver um mapeamento centrado, e na figura A1.3 o mapeamento segundo a origem.

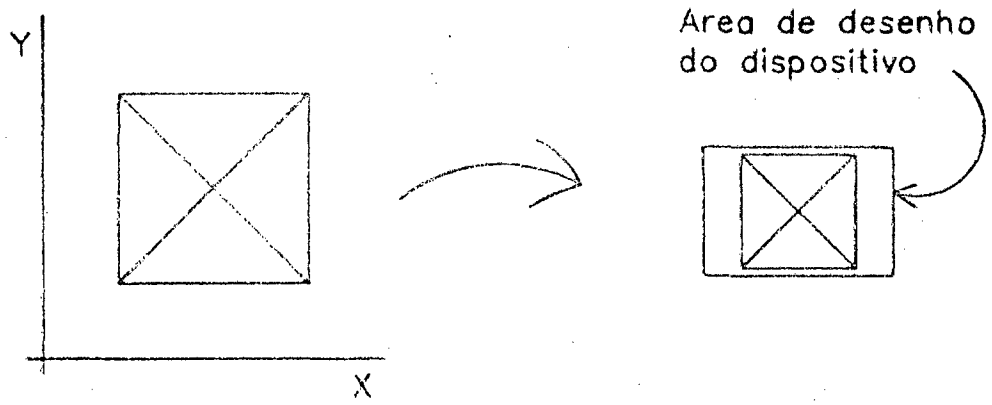


Figura A1.2 - O mapeamento centrado.

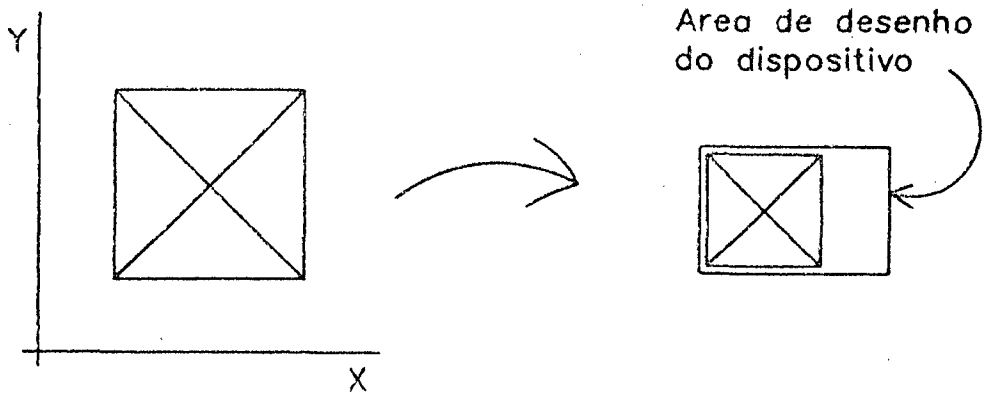


Figura A1.3 - O mapeamento na origem.

Observa-se que para realizar o enquadramento, deve-se calcular a escala que transforma as maiores dimensões do objeto de modo a ajustar-se ao tamanho da área de desenho do dispositivo utilizado para a representação. A escala é então escolhida em função das dimensões da *window* e do *viewport*. Devem ser calculadas as escalas em relação aos eixos X e Y. A menor delas deve ser escolhida. Nota-se que o *fator de aspecto*, ou relação

entre as resoluções vertical e horizontal do dispositivo, deve ser considerado no cálculo da escala para o eixo Y. O cálculo do *fator de aspecto*, aqui representado como "ASPEC", está detalhado a seguir.

$$ASPEC = (ymax / xmax) * (lx / ly) \qquad (A1.1)$$

onde

(ymax / xmax) é a relação entre o número máximo de pontos nos eixos X e Y. Nos vídeos padrão IBM PC esta relação tem o valor de (200/640) = 0.3125 em alta resolução, e (200/320) = 0.6250 em média resolução.

(lx / ly) é a relação entre as maiores dimensões vertical e horizontal do dispositivo. Nos vídeos padrão IBM PC temos geralmente (lx / ly) = 1.33.

Desta maneira, o fator de aspecto para os vídeos acima citados é geralmente

$$ASPEC = 0.3125 * 1.33 = \mathbf{0.4156}$$

Observando as coordenadas definidas na figura A1.4, pode-se definir o cálculo da escala para enquadramento do desenho.

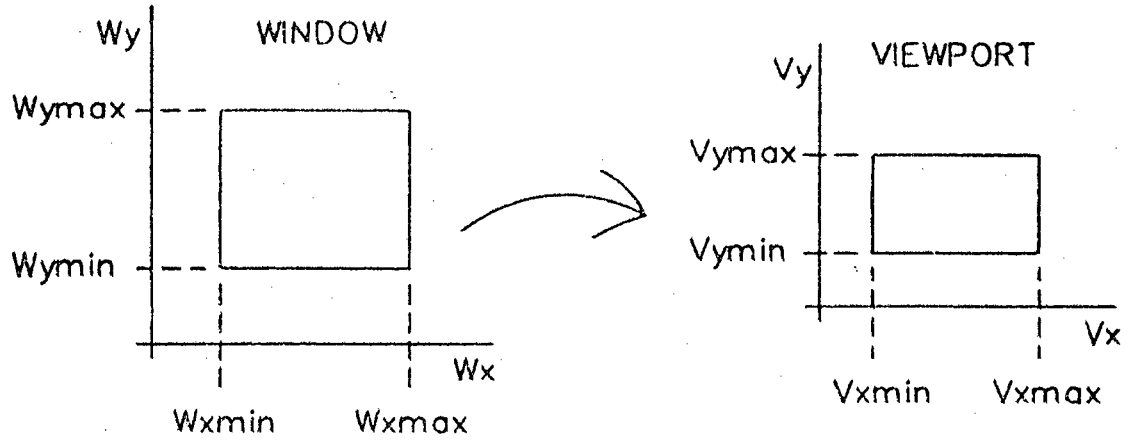


Figura A1.4 - Mapeamento de coordenadas.

A escala "ESC" é calculada para os eixos X e Y, resultando nas escalas escX e escY. A escala escolhida será então a menor delas, de modo a possibilitar o completo enquadramento do desenho segundo o eixo da escala escolhida. O desenho será centrado na direção da outra escala, conforme a figura A1.2. No cálculo da escala em Y, deve-se considerar o *fator de aspecto* para transformar o valor calculado para Y em valores coerentes com o valor calculado para a direção X, de modo a permitir uma correta comparação entre escX e escY.

$$escX = (Vxmax - Vxmin) / (Wxmax - Wxmin) \quad (A1.2)$$

$$escY = ((Vymax - Vymin) / ASPEC) / (Wymax - Wymin) \quad (A1.3)$$

$$ESC = min(escX,escY) \quad (A1.4)$$

Para o caso do vídeo, no Editor Gráfico, optou-se pela representação centrada do objeto na tela. Este desenvolvimento pode ser utilizado para qualquer vídeo do tipo *raster*, entrando-se com os valores adequados para o tamanho do *viewport*, e o fator de aspecto.

Considerando um ponto de coordenadas (Wx,Wy) definido na *window*, ou seja, no sistema de coordenadas do usuário, seu correspondente (Vx,Vy) definido no *viewport* ou sistema de coordenadas do dispositivo, e (Wxmed,Wymed) e (Vxmed,Vymed) como os pontos médios da *window* e do *viewport* respectivamente, tem-se :

$$Wxmed = (Wxmax + Wxmin) / 2$$
$$Wymed = (Wymax + Wymin) / 2 \qquad (A1.5)$$
$$Vxmed = (Vxmax + Vxmin) / 2$$
$$Vymed = (Vymax + Vymin) / 2$$

Então,

$$(Wx - Wxmed) * ESC = (Vx - Vxmed) \qquad (A1.6)$$

de onde $$Vx = Vxmed + (Wx + Wxmed) * ESC \qquad (A1.7)$$

De maneira similar podemos calcular a transformação para o eixo Y, onde deve-se novamente considerar o problema do fator de aspecto "ASPEC".

$$(Wy - Wymed) * ESC * ASPEC = (Vy - Vymed) \quad (A1.8)$$

de onde $$Vy = Vymed + (Wy + Wymed) * ESC * ASPEC \qquad (A1.9)$$

No caso da impressora e do traçador gráfico, optou-se pelo mapeamento segundo a origem. Conforme a figura A1.4, e utilizando as definições anteriores para a escala "ESC" e para p *fator de aspecto* do dispositivo "ASPEC", tem-se

$$(Wx - Wxmin) * ESC = Vx - Vxmin \qquad (A1.10)$$

de onde $$Vx = Vxmin + (Wx - Wxmin) * ESC \qquad (A1.11)$$

E, de maneira similar,

$$(Wy - Wymin) * ESC * ASPEC = Vy - Vymin \quad (A1.12)$$

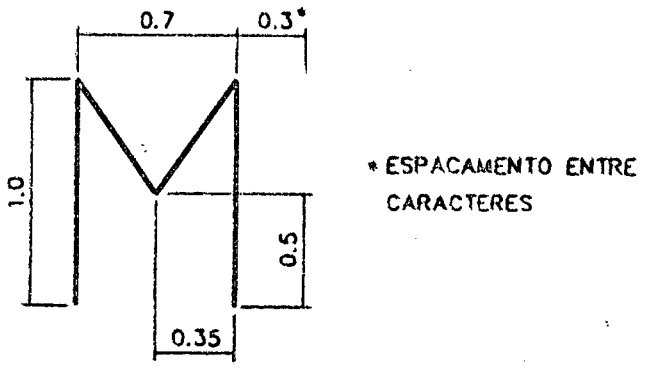
de onde $$Vy = Vymin + (Wy - Wymin) * ESC \qquad (A1.13)$$

A1.2. A definição do texto

A forma mais comum para a definição de caracteres de texto em dispositivos gráficos é em forma de matriz de pontos. Este método, apesar de bastante rápido, não satisfaz às necessidades de um editor gráfico, principalmente pela dificuldade de se realizar um escalamento ou rotação sobre a matriz de pontos. Além disso, a matriz de caracteres torna o texto dependente do dispositivo, em função das diferentes resoluções e fatores de aspecto encontrados.

O Editor Gráfico utiliza uma forma vetorial para a repre- re-
sentação dos caracteres de texto. Esta forma permite inclusive uma interface muito mais simples quando o dispositivo utilizado é do tipo vetorial, como os traçadores gráficos, por exemplo.

Cada caracter é armazenado como uma série de incrementos nas direções X e Y. A partir do ponto de origem do caracter (geralmente o canto inferior esquerdo), são definidos os incre-



EGG.TXT => 0,0,0,1,0.35,-0.5,0.35,0.5,0,-1

Figura A1.5 - O caracter de texto.

mentos nas duas direções, e os pontos são unidos por retas, formando o caracter. Os caracteres seguem a forma básica de um retângulo de altura unitária. Os incrementos e a largura do caracter são frações da sua altura. Desta maneira torna-se possível a representação do caracter em qualquer tamanho, bastando multiplicar a altura pelo tamanho desejado.

Na figura A1.5 pode-se ver o exemplo do caracter "M", mostrando suas dimensões e os dados armazenados no arquivo "EGG.TXT" para sua representação.

Utilizando este tipo de representação, pode-se desenhar o texto em qualquer dispositivo, pois o mapeamento *window-viwport* fica responsável pela proporcionalidade dos caracteres, além de se ter uma facilidade para operações de transformação sobre o texto, como escalamento ou rotação, bastando aplicar as matrizes de transformação sobre os incrementos de cada caracter.

A1.3. Traçado da linha em dispositivos tipo *raster*

Existem muitos algoritmos que produzem um bom resultado quanto ao aspecto da linha traçada [18,19,20,22]. Entretanto, deve-se levar em conta o tempo gasto para executar a tarefa. Os algoritmos que utilizam instruções de multiplicação ou divisão, exponenciação, arredondamento ou outras instruções que consomem bastante tempo são geralmente mais lentos e devem ser evitados. Foi utilizado o algoritmo *integer DDA (Integer Digital Differential Analyzer)*, ou DDA inteiro [18], que utiliza so-

mente inteiros, não necessitando de nenhuma instrução de ponto flutuante, sendo por isso mesmo bastante rápido. Além de rápido, este algoritmo fornece uma linha de bom aspecto.

O algoritmo é dividido em quatro casos, dependendo da inclinação da linha. Para que o número de casos seja reduzido a quatro, trabalha-se somente com linhas crescentes na direção Y, ou seja, deltaY = (Yfinal - Yinicial) positivo. Caso deltaY seja negativo, a procedure *swap* é chamada, invertendo a posição dos pontos inicial e final, e corrigindo também os valores de deltaX e deltaY (trocam de sinal).

Uma variável "erro" é utilizada para a determinação da direção do incremento para o próximo ponto a ser plotado. Esta variável é inicializada com zero, e volta a tornar-se nula no fim do traçado ou cada vez que o ponto plotado coincidir com a linha real, ou seja, indica a distância do ponto plotado à linha real. A análise do sinal do "erro" indicará em que direção será plotado o próximo ponto. Os quatro casos possíveis estão descritos a seguir.

Caso 1 - Inclinação entre 0 e 1

* erro < 0 - plota o ponto uma unidade à direita do ponto anterior, e soma deltaY ao erro.

* erro > = 0 - plota o ponto uma unidade acima e uma unidade à direita do ponto anterior, e soma (deltaY - deltaX) ao erro.

Caso 2 - Inclinação maior que 1

* erro < 0 - plota o ponto uma unidade acima e uma unidade à direita do ponto anterior, e soma (deltaY - deltaX) ao erro.

* erro > = 0 - plota o ponto uma unidade acima do ponto anterior, e subtrai deltaX do erro.

Caso 3 - Inclinação entre -1 e 0

* erro < 0 - plota o ponto uma unidade à esquerda do ponto anterior, e soma deltaY ao erro.

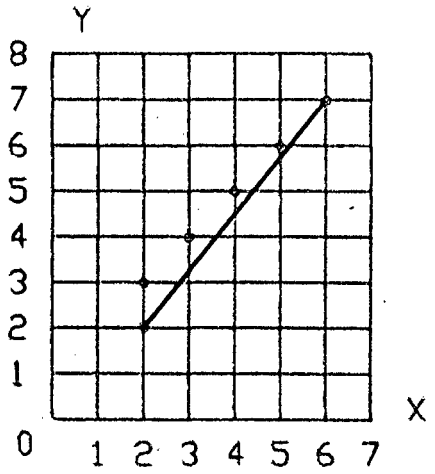
* erro > = 0 - plota o ponto uma unidade acima e uma unidade à esquerda do ponto anterior, e soma (deltaY + deltaX) ao erro.

Caso 4 - Inclinação menor que -1

* erro < 0 - plota o ponto uma unidade acima e uma unidade à esquerda do ponto anterior, e soma (deltaY + deltaX) ao erro.

* erro >= 0 - plota o ponto uma unidade acima do ponto anterior, e soma deltaX do erro.

A seguir tem-se um exemplo de uma linha traçada pelo algoritmo DDA inteiro, com as iterações mostradas passo a passo, conforme a linha da figura A1.6.



xinicio := 2 xfinal := 6

yinicio := 2 yfinal := 7

deltaX := 4 deltaY := 5

deltaY - deltaX := 1

Figura A1.6 - O traçado da linha.

Iteração

1	erro = 0 x = 2 y = 2+1 = 3 erro = 0-4 = -4
2	erro = -4 x = 2+1 = 3 y = 3+1 = 4 erro = -4+1 = -3
3	erro = -3 x = 3+1 = 4 y = 4+1 = 5 erro = -3+1 = -2
4	erro = -2 x = 4+1 = 5 y = 5+1 = 6 erro = -2+1 = -1

Os pontos inicial e final da linha são plotados separadamente, pois são dados de entrada, não havendo a necessidade do cálculo de suas posições.

A1.4. Traçado do círculo em dispositivos *tipo raster*

Para o traçado do círculo em dispositivos do tipo *raster* [3,18,19,20,22] é utilizada geralmente a sua representação polar, onde cada ponto pertencente ao círculo tem sua posição definida por

$$\begin{aligned} x &= x_C + R\cos(\theta) \\ y &= y_C + R\sin(\theta) \end{aligned} \qquad \text{(A1.14)}$$

onde θ é dado em radianos, variando de 0 a 2π. Deste modo, as expressões (A1.14) devem ser calculadas para cada ponto plotado, o que resulta num tempo muito grande para o traçado completo do círculo. Utilizam-se então alguns procedimentos para reduzir o tempo de cálculo dos pontos.

Uma redução considerável no tempo gasto para o traçado do círculo pode ser conseguido utilizando-se o método do desenho incremental, que calcula cada ponto a partir do anterior. Assim, o seno e o coseno, que são instruções que consomem um tempo bastante grande para sua execução, são executadas apenas uma vez no cálculo do círculo. Para ilustrar este procedimento, considere-se um círculo centrado na origem. Desta maneira, os termos x_C e y_C das expressões (A1.14) são nulos, e dois pontos consecutivos do círculo podem ser representados por

$$\begin{aligned} x_1 &= R\cos(\theta) \\ y_1 &= R\sin(\theta) \\ x_2 &= R\cos(\theta+d\theta) \\ y_2 &= R\sin(\theta+d\theta) \end{aligned} \qquad \text{(A1.15)}$$

onde $d\theta$ é o incremento angular, constante. Por trigonometria,

$$x_2 = R\cos(\theta)\cos(d\theta) - R\sin(\theta)\sin(d\theta)$$
$$y_2 = R\sin(\theta)\cos(d\theta) - R\cos(\theta)\sin(d\theta) \qquad (A1.16)$$

Substituindo os valores de x_1 e y_1 das expressões (A1.15),

$$x_2 = x_1\cos(d\theta) - y_1\sin(d\theta)$$
$$y_2 = y_1\cos(d\theta) + x_1\sin(d\theta) \qquad (A1.17)$$

Com estas expressões pode-se calcular cada ponto do círculo a partir do anterior, sendo necessário o cálculo de cos(dθ) e sin(dθ) apenas no início do algoritmo.

Um fato que também deve ser aproveitado é a simetria do círculo. pode-se calcular apenas os pontos que se encontram sobre um oitavo do círculo, ou seja, para um intervalo de 45°. A partir deste ponto outros sete podem ser posicionados por simetria. Se o ponto de coordenadas (a,b) pertence ao círculo, conforme figura A1.7,os pontos (-a,b), (a,-b), (-a,-b), (b,a), (-b,a), (b,-a), e (-b,-a) também pertencem ao mesmo círculo. Reduz-se assim em oito vezes o número de pontos a serem calculados para o traçado completo do círculo, ganhando-se desta forma em velocidade.

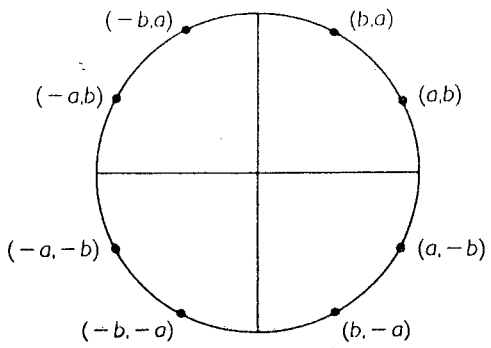


Figura A1.7 - A simetria no círculo.

Para reduzir ainda mais este tempo, pode-se desenhar o círculo com pequenos segmentos de reta ou linhas. Deste modo pode-se aumentar o incremento angular dθ, necessitando-se de menos pontos calculados para o mesmo círculo. Quanto menor o número de segmentos utilizados, menor o número de pontos calculados e, portanto, menor o tempo gasto para o desenho do círculo. Entretanto, a redução do número de segmentos de reta utilizados provoca uma redução da qualidade do desenho. Deve-se escolher um número de pontos que seja o menor possível, de modo que no desenho do círculo não se possa distinguir cada segmento de reta que agora o compõe. O algoritmo implementado no Editor Gráfico utiliza sete segmentos de reta para cada oitava parte do círculo, ou seja, um total de 56 segmentos.

APENDICE 2

EXEMPLO DO ARQUIVO GRAFICO

Os dados abaixo são os componentes do arquivo gráfico do desenho mostrado na figura A2.1. O desenho é constituído de um exemplo de cada elemento básico de criação do Editor Gráfico. Os comentários colocados após os asteriscos não fazem parte do arquivo, aparecendo apenas para um melhor entendimento do formato do mesmo. O desenho foi feito utilizando dois níveis, e dois tipos de linha diferentes. No primeiro nível estão todos os elementos básicos de criação, como aparecem na figura A2.2. No nível 2 aparecem o texto "TEXTO INSERIDO NA FOLHA 2" e uma linha tracejada, de acordo com a figura A2.3.

```
ARQUIVO ==> EXEMPLO * linha de comentários (nome do arquivo)
23.7205  14.2000  8.5000 * escala e ponto central
 110001.0000          * linha contínua (atributo 0)
      1.0000
      1.0000
     25.0000
      1.0000
      1.0000
      1.0000
     25.0000
      1.0000
 120002.0000          * linha poligonal
      2.0000
      5.0000
```

```
     12.0000
      8.0000
          11
      2.0000
      5.0000
      3.0000
      8.0000
      4.0000
      5.0000
      5.0000
      8.0000
      6.0000
      5.0000
      7.0000
      8.0000
      8.0000
      5.0000
      9.0000
      8.0000
     10.0000
      5.0000
     11.0000
      8.0000
     12.0000
      5.0000
130003.0000          * retângulo
      3.0000
     10.0000
     10.0000
     14.0000
      3.0000
     10.0000
     10.0000
     10.0000
     10.0000
     14.0000
      3.0000
     14.0000
140004.0000          * polígono
     17.0000
      5.0000
     23.0000
     11.0000
           8
     19.0000
      5.0000
     21.0000
      5.0000
     23.0000
      7.0000
     23.0000
      9.0000
     21.0000
     11.0000
     19.0000
     11.0000
```

```
     17.0000
      9.0000
     17.0000
      7.0000
150005.0000          * círculo
     18.0000
      6.0000
     22.0000
     10.0000
     20.0000
      8.0000
      2.0000
180006.0000          * arco
     12.0000
      9.0000
     19.0000
     14.0000
     19.1364
      6.4091
      7.5921
     91.0292
    160.0462
190007.0000          * texto
      8.0000
      2.0000
     18.0000
      4.0000
TXT TEXTO
      8.0000
      2.0000
      2.0000
      0.0000
290008.0000          * texto inserido na folha 2
      3.0000
     15.0000
     27.4000
     16.0000
TXT TEXTO INSERIDO NA FOLHA 2
      3.0000
     15.0000
      1.0000
      0.0000
20210009.0000        * linha tracejada inserida na folha 2
     25.0000
      3.0000
     25.0000
     13.0000
     25.0000
      3.0000
     25.0000
     13.0000
```

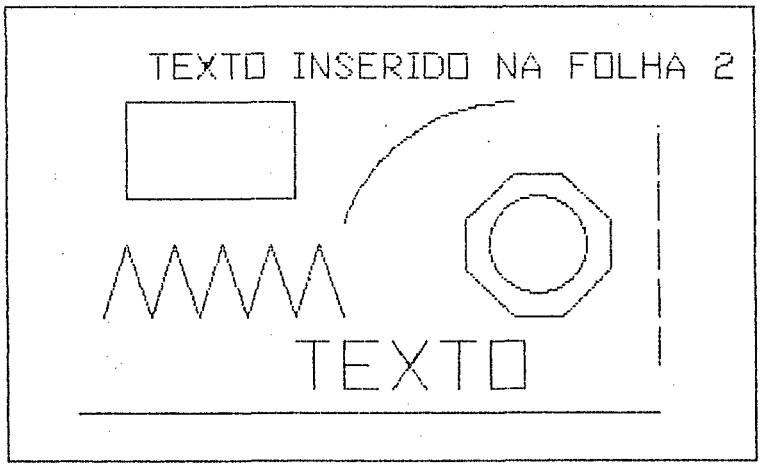


Figura A2.1 - O desenho completo.

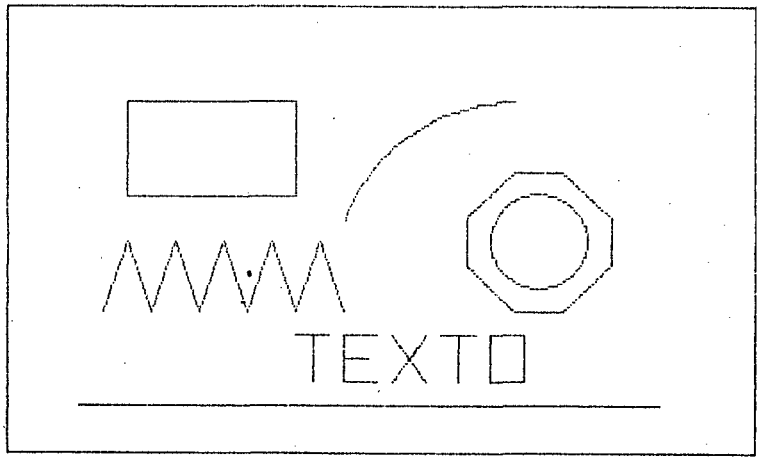


Figura A2.2 - O nível 1 do desenho.

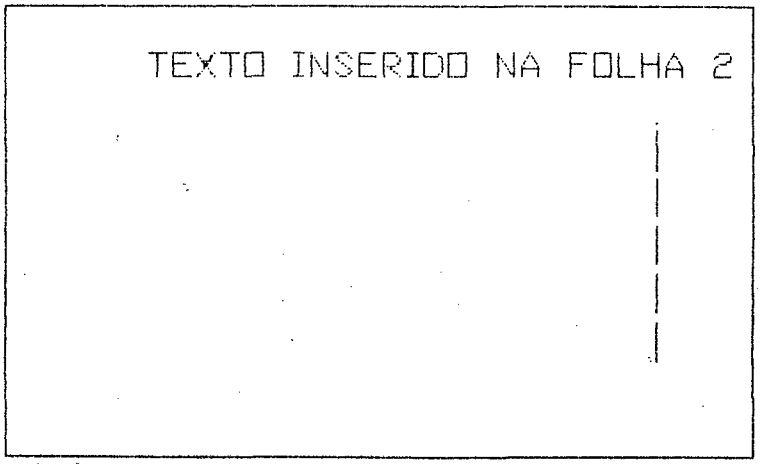


Figura A2.3 - O nível 2 do desenho

Um arquivo de bloco (extensão ".BLO") possui a mesma especificação que o arquivo de desenho. As linhas iniciais é que apresentam uma diferença. A primeira linha contém, como no arquivo de desenho, o nome do bloco. A segunda e a terceira linha

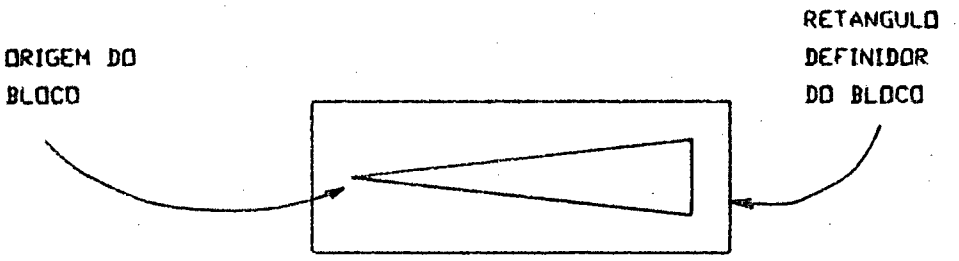


Figura A2.4 - O bloco definido.

contém as coordenadas dos quatro vértices do retângulo que define o bloco. Exemplificando com o desenho da figura A2.4, composto de um único elemento (polígono) definido pelas coordenadas indicadas, e com o bloco definido pelo retângulo tracejado, ao ser dado o comando *bsalva* com o nome "seta", é criado o arquivo "SETA.BLO", cujo conteúdo será o seguinte

```
BLOCO ==> SETA                  * linha de comentários (nome do bloco)
-0.5000  -1.2500  8.5000  -1.2500 * coordenadas que definem
 8.5000   1.7500  -0.5000  1.7500  *  o  bloco
  40001.0000                        * definição do elemento polígono
      0.0000
     -1.0000
      8.0000
      1.0000
      3
      0.0000
      0.0000
      8.0000
      1.0000
      8.0000
     -1.0000
```

No cabeçalho de cada elemento no arquivo de bloco não há a definição de folha (ou nível) de desenho. Quando o bloco é inserido em algum desenho, cada elemento é definido como pertencente à folha ativa do momento da inserção.

APENDICE 3

PROTOCOLOS DE COMUNICAÇAO

A3.1. Traçador gráfico (*plotter*)

Com o objetivo de prover uma saída gráfica rápida e eficiente para o Editor Gráfico, desenvolveu-se o *driver* para o traçador gráfico TDD 21R da Digicon [51]. Um *driver* é um programa que estabelece um protocolo de comunicação entre dois dispositivos ou softwares, ou seja, permite que os dados contidos no arquivo gráfico de um desenho sejam interpretados de modo a gerar este desenho em papel no traçador.

O modelo TDD 21R da Digicon permite o trabalho com formatos de folha tamanho A1 e A2, troca de penas manual, e é compatível com a linha DMP da Houston Instruments. Isto significa que softwares que oferecem *drivers* para esta linha funcionam com o TDD 21R.

O TDD 21R possui um microprocessador capaz de interpretar comandos enviados pelo microcomputador. Isto significa que o traçador possui um conjunto de instruções internas para a realização de operações como o traçado de um círculo, de um arco, texto, etc. Deste modo, não é necessária a elaboração de algoritmos para o traçado das primitivas geométricas. Pode-se enviar ao traçador a instrução ou comando que realiza aquela operação seguida dos parâmetros correspondentes.

Os comandos gráficos, ou seja, os dados e instruções são enviados do microcomputador ao traçador via interface serial padrão RS 232-C, utilizando o formato ASCII. Desta forma, qualquer linguagem de alto nível, como Pascal ou C, por exemplo, pode ser utilizada para a elaboração de um *driver*. Os dados e comandos são enviados por meio de uma instrução de I/O para saída, como a instrução

```
write(aux,':: T');
```

do Pascal, que habilita o traçador a receber dados do microcomputador (":"), e aciona o gráfico de teste ("T").

Estes dados devem ser direcionados para uma das portas de comunicação serial reconhecidas pelo Sistema Operacional, ou seja, COM1 ou COM2. A porta serial utilizada deve ser configurada segundo os padrões do traçador, ou seja, taxa de transferência de 9600 baud, paridade ímpar, 8 bits de dados e um stop-bit. Estes parâmetros podem ser ajustados com o comando MODE do Sistema Operacional.

Alguns comandos do traçador devem ser enviados antes dos comandos gráficos. Inicialmente deve-se preparar o traçador para trabalho, ou seja, habilitá-lo para receber dados do microcomputador, com o comando ":". O traçador permanece habilitado até que um comando de desabilite ("@") ou de reset ("Z") seja enviado.

O TDD 21R trabalha com duas orientações dos eixos coordenados, de acordo com o formato de papel escolhido (A1 ou A2), como pode ser visto na figura A3.1. O formato será selecionado de acordo com o tamanho da área de desenho especificada pelo usuário. A partir da área de desenho especificada são selecionados o formato e o campo de trabalho. A origem do desenho no

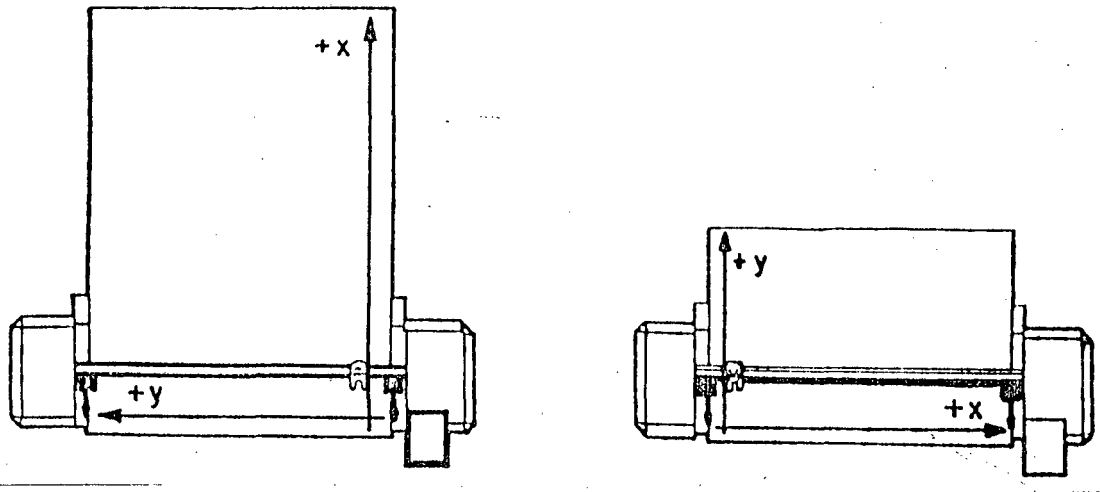


Figura A3.1 - A orientação dos eixos e a origem absoluta.

papel também deve ser especificada pelo usuário e é enviada para o traçador em relação à origem absoluta do sistema de coordenadas selecionado (função do formato).

Para se obter o melhor traçado possível, deve-se escolher uma velocidade de trabalho adequada ao tipo de papel e tipo de pena utilizado. Esta velocidade, indicada pelo usuário, deve ser selecionada através do comando correspondente. A resolucão endereçável deve ser também selecionada. Neste caso selecionou-se a resolução de 0,1mm.

Depois do ajuste e seleção destes parâmetros, pode-se iniciar o envio das instruções gráficas. O arquivo gráfico, que contêm os dados referentes ao desenho, é varrido e cada elemento geométrico tem seus parâmetros multiplicados por uma escala de modo que o desenho no papel se conforme com o tamanho desejado pelo usuário. Estes parâmetros são então enviados ao traçador juntamente com o comando correspondente aquele elemento.

Para exemplificar, toma-se uma linha cuja definição no arquivo gráfico (veja capítulo 3) tem o seguinte formato

```
110001.00
   10.00
   20.00
  500.00
  600.00
   10.00
   20.00
  500.00
  600.00
```

Considerando que os valores acima estão em milímetros, e a resolução selecionada do traçador é de 0,1mm, deve-se aplicar uma escala de 10:1 aos parâmetros do elemento. Os comandos e dados necessários para o desenho da linha no traçador são os seguintes

U	(levanta a pena)
100, 200	(deslocamento até as coordenadas dadas)
D	(abaixa a pena)
5000, 6000	(desloca até o ponto dado)
U	(levanta a pena)

Estes dados e comandos poderiam ser codificados numa instrução em Pascal do tipo

```
write(aux,'U 100,200 D 5000,6000 U');
```

No caso de um círculo com a seguinte representação no arquivo gráfico

```
150002.00
    50.00
    50.00
   150.00
   150.00
   100.00
   100.00
    50.00
```

deve-se enviar para o traçador o seguinte comando

```
CC 1000,1000 500 (traça o círculo e levanta a pena)
```

que pode ser codificado em Pascal como

```
write(aux,'CC 1000,1000 500');
```

Assim sendo, cada elemento do arquivo gráfico é interpretado e uma instrução em Pascal é gerada e executada, de forma a enviar todos os dados para o traçador, e o desenho é todo gerado no papel. Na tabela A3.1 é apresentado um resumo dos comandos do traçador gráfico TDD 21R.

COMANDO	DESCRIÇÃO
;:	Seleciona o traçador para comunicação
@	Desabilita a comunicação
Z	Reset
T	Gráfico de teste
U	Levanta a pena
D	Abaixa a pena
EH	Formato de folha pequeno (A2)
EF	Formato de folha grande (A1)
Vn	Seleção de velocidade (n = 0 a 255)
ECn	Seleção da resolução endereçável, sendo n = M 0,1 mm N 0,025 mm 1 0.001" 5 0.005"
H	Origem absoluta
O	Nova origem
A	Posicionamento absoluto
R	Posicionamento relativo
x, y	Deslocamento vetorial
Pn	Nova pena
CC x, y r	Círculo, sendo x, y centro do círculo r raio
CA x, y g	Arco, sendo x, y centro do arco (em incrementos) g tamanho do arco, em graus
SraaCHR(n)_	Texto, onde r indica a direção da escrita aa altura dos caracteres CHR(n) texto _ indica fim do comando

Tabela A3.1 - Comandos do TDD 21R.

A3.2. Impressora matricial

Uma maneira mais simples e de baixo custo, porém não tão eficiente, para se obter uma saída gráfica é a utilização de impressoras matriciais. Relativamente ao traçador, as impressoras deixam muito a desejar quanto à qualidade da impressão. Além disso, o tamanho da área de desenho é limitado pelo tamanho do papel utilizado (formulário contínuo) que tem no máximo 132 colunas, ou seja, aproximadamente 335 x 280 mm. Apesar destes fatores, o custo de uma impressora matricial é bem inferior ao de um traçador gráfico.

O *driver* para impressora do Editor Gráfico foi desenvolvido para as impressoras Grafix [3,52], compatíveis com a impressora Epson, talvez a mais popular impressora matricial do mercado mundial, principalmente Norte-Americano. Pode-se configurar o *driver* para o trabalho com impressoras tanto de 80 como 132 colunas.

Devido ao fato da impressora matricial ser um dispositivo do tipo *raster* e não vetorial, como é o caso do traçador gráfico, não há a possibilidade de interpretar cada elemento do arquivo gráfico e imediatamente enviá-lo à impressora. É necessária a conversão vetorial - raster, chamada "rasterização", com a criação de uma *página gráfica* na memória RAM do microcomputador. O desenho deve ser interpretado e desenhado nesta página de memória, para então ser enviado, ponto a ponto, para a impressora. O tamanho deste espaço de memória

deve ser dimensionado de acordo com a área útil de desenho, expressa em pontos endereçáveis.

Foi selecionado o modo de impressão em densidade dupla da Grafix. Este modo de impressão fornece uma resolução de 120 pontos/polegada na horizontal e 72 pontos/polegada na vertical. Como cada ponto necessita de apenas 1 bit na memória para ser armazenado (ligado ou desligado), pode-se armazenar 8 pontos em cada byte.

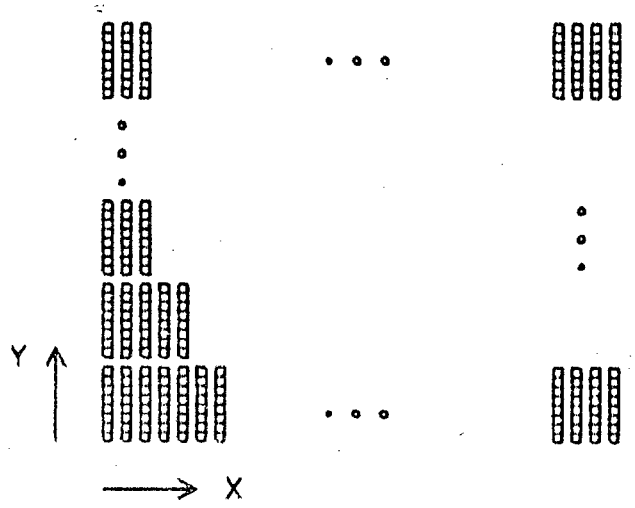


Figura A3.2 - A página gráfica.

Na figura A3.2 pode-se observar um esquema da página gráfica, onde cada retângulo composto de oito espaços representa um byte (cada espaço de retângulo é um bit). Utilizando a memória desta maneira, deve-se dimensionar a página gráfica com um número de colunas igual ao comprimento do desenho, em pontos, e o número de linhas igual à altura do desenho, em pontos, divi-

dido por oito. Como a impressora no modo gráfico imprime com oito agulhas dispostas em linha no sentido vertical, enviando-se para a mesma cada byte de uma linha da página gráfica tem-se oito linhas de pontos impressas no papel.

Para endereçar um ponto de coordenadas X,Y na página gráfica, deve-se seguir o procedimento mostrado a seguir. Seja a matriz *pagina* dimensionada da seguinte forma

```
var pagina : array [1..col_max,1..lin_max] of byte;
```

onde col_max = maior coordenada na direção X

lin_max = maior coordenada na direção Y / 8 + 1

e o vetor *bits* definido como a seguir

```
const bits : array [1..7] of byte = (1,2,4,8,16,32,64,128);
```

e utilizando ainda as variáveis *i,j,k*, declaradas como inteiros, para indicar os componentes da matriz *pagina[i,j]* (índices), e do vetor *bits[k]*, tem-se, para um ponto de coordenadas X,Y a ser setado, as seguintes etapas

```
i:= X;
j:= Y div 8 + 1;     {divisão inteira}
k:= Y mod 8;         {resto da divisão inteira}
pagina[i, j]:= pagina[i, j] OR bits[k];
```

O valor de *j*, ou a linha da matriz *pagina* é calculada lada pela divisão inteira (função *div* do Pascal) do valor de Y por oito, mais uma unidade. O resto desta divisão (encontrado pela função *mod* do Pascal) retorna o bit (valor de) a ser setado no byte designado por *pagina[i,j]*. Para setar este bit, utiliza-se a função lógica *or* do conteúdo do próprio byte com o valor contido na posição *I* do vetor *bits[k]*, de modo que os bits que

já haviam sido setados naquele byte sejam mantidos. Assim, por exemplo,

(01000100) OR (00000010) = (01000110)

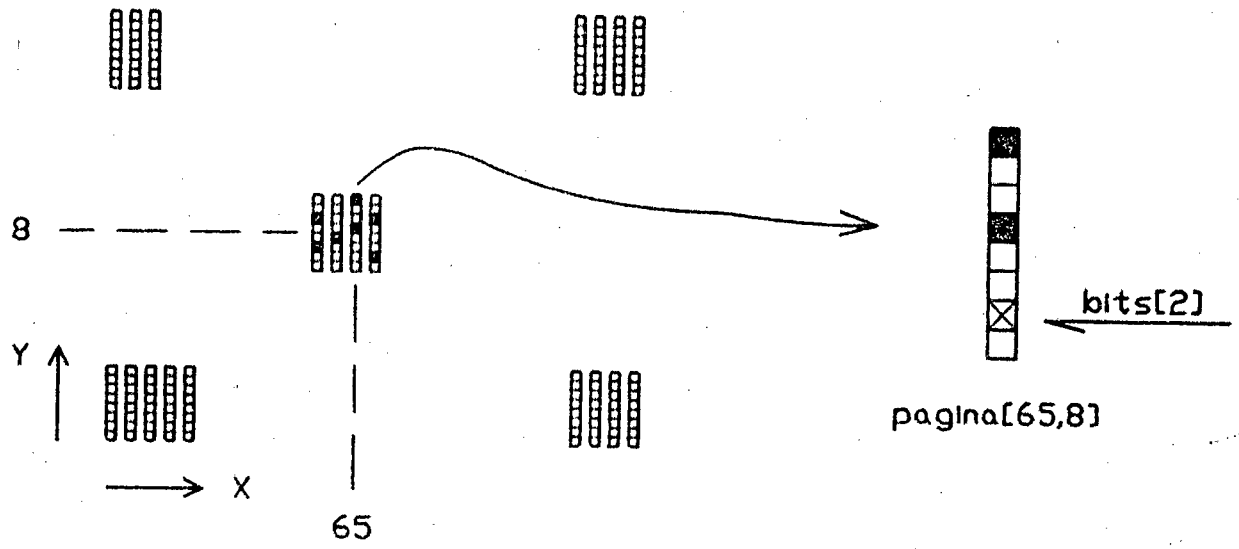


Figura A3.3 - Exemplo do ponto setado na memória.

Utilizando o exemplo da matriz mostrada na figura A3.3, admitindo alguns pontos já setados, o procedimento para setar o ponto de coordenadas (65,58) é o seguinte

```
i:= 65;
j:= 58 div 8 + 1 = 8;
k:= 58 mod 8 = 2;
pagina[65,8]:= pagina[65,8] or bits[2];
```

que corresponde a

(10010000) or (00000010) = (10010010)

A geração da página gráfica é feita a partir do arquivo gráfico. Cada elemento do arquivo é interpretado e seu desenho é gerado em pontos na memória, ou seja, na página gráfica. Os algoritmos utilizados para a geração dos desenhos na memória são os mesmos que geram estes desenhos na tela. Apenas há um direcionamento dos pontos para a memória principal ao invés da memória de vídeo, e o *fator de aspecto* da impressora é agora utilizado.

Para a impressão da página gráfica, deve-se primeiramente selecionar o modo gráfico da impressora. A instrução em Pascal que seleciona o espaçamento entre linhas de 8/72 pontos por polegada (resolução vertical) é o seguinte

```
write(lst, chr(27)'A'#8);
```

A resolução horizontal é enviada a cada linha para a impressora, juntamente com a informação do tamanho da linha que vai ser impressa. Por este motivo, a cada linha deve-se determinar o último ponto setado. A sequência de instruções abaixo pode ser utilizada para a impressão da página gráfica.

```
for i:= 1 to lin_max do begin
  ultimo:= col_max;
  while pagina[ultimo, i] = 0 do ultimo:= ultimo -1;
  write(lst, chr(27)'L', chr(lo(ultimo)), chr(hi(ultimo)));
end;
```