UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA DE CAMPINAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

# INTERFACE DE SAÍDA PARA UM SISTEMA GRÁFICO DE BASE

POR : Luis Fernando Ramos Molinaro
ORIENTADOR: Prof.Dr. Mario Jino

Tese de mestrado apresentada a Faculdade de Engenharia de Campinas da Universidade Estadual de Campinas.

NOVEMBRO - 1981

AGRADECIMENTOS

Agradeço de forma especial ao meu orientador Prof. Dr. Mario Jino pela ajuda no desenvolvimento desse trabalho e ao colega Roberto Pinto Martins pelas discussões elucidativas e cooperação que foram de inestimável valor.

Finalmente, um agradecimento a todos que de alguma forma colaboraram para o êxito desse trabalho.

A meus pais pelo apoio,
carinho e dedicação

"Quando um dia acontecer (como certamente acontecerá) serem notadas imperfeições tais a êste estudo, que resulte declarada a sua inutilidade, apenas ficarei lamentando não ter sabido dar-lhe aquêle mínimo valor que assegurasse a sobrevivência desta página, - única, afinal, em cuja sobrevivência me empenhava"

Antônio Salgado Júnior

# SUMÁRIO

O objetivo desta tese é a definição e o desenvolvimento de uma interface de saída para um sistema gráfico de base, tendo como núcleo o GKS. O GKS ("Graphical Kernel System") é uma proposta para a padronização de uma linguagem gráfica; esta linguagem aceita a descrição da geometria de objetos no espeço bidimensional, os quais podem ser agrupados para fim de manipulação numa entidade denominada segmento. Uma característica peculiar desta proposta é o conceito de "workstation" que, considerada como uma unidade lógica, engloba uma superfície de visualização e diversos dispositivos de entrada.

Para atingir-se o objetivo preconizado, podemos dividir este trabalho em três fases: na primeira, são definidas todas as características da interface, ou seja, como ocorre a comunicação do sistema gráfico de base com os diversos dispositivos gráficos de saída; na segunda, é apresentado um modelo funcional para a interface; e na terceira, é efetuada a aplicação do modelo na implementação de uma interface para um vídeo vetorial regenerativo.

# ÍNDICE

# CAPÍTULO 1

# INTRODUÇÃO

Sistemas gráficos são reconhecidos como elementos vitais na comunicação entre o homem e a máquina. Eles permitem a troca de informação gráfica entre o usuário e o sistema computacional por meio de dispositivos gráficos de saída e entrada. Sistemas gráficos fazem parte de um universo mais amplo que é a computação gráfica, conjunto de metodologias envolvendo a geração, representação, manipulação ou cálculo de objetos gráficos por computador, bem como a sua associação com informação de caráter não gráfico com eles relacionados /1/.

Sistemas gráficos variam consideravelmente em capacidade e generalidade. Num extremo, um sistema pode ser projetado em torno de um dispositivo específico, cujas características podem ser totalmente exploradas; a principal desvantagem de um sistema deste tipo é consequência da inflexibilidade dos programas de aplicação desenvolvidos, os quais poderão tornar-se inúteis caso o dispositivo gráfico existente seja substituído por outro de características diferentes. No outro extremo, temos os sistemas independentes de dispositivos. Embora a utilização destes sistemas independentes possa causar a subutilização das potencialidades de alguns tipos de dispositivos, esta independência isola o programa de aplicação das particularidades dos diferentes tipos de dispositivos, dotando-o de um alto grau de portabilidade.

Sistemas gráficos de propósitos gerais independentes de dispositivos, os quais passaremos a denominar sistemas gráficos de base, fazem parte do universo dos sistemas independentes de dispositivos e caracterizam-se por disporem de um conjunto de capacidades gráficas simples e fundamentais. Para incentivar a utilização de sistemas gráficos, estão sendo definidas linguagens gráficas de propósitos gerais para os sistemas gráficos de base, visando padronizá-las com o objetivo de obter vantagens econômicas no desenvolvimento e uso de sistemas gráficos. Dentre os vários trabalhos para a padronização de uma linguagem gráfica de propósitos gerais, inclui-se o GKS ("Graphical Kernel System") apresentado pelo NI ("normung institute") /6/. Neste trabalho define-se um núcleo de um sistema gráfico de base, cuja principal vantagem é a portabilidade do programa de aplicação, alcançada pela padronização da interface entre o programa de aplicação e o sistema gráfico de base.

Está sendo implementado na UNICAMP um sistema gráfico de base, tendo como núcleo o GKS, com vistas a implantação de um sistema de processamento de informação gráfica para controle de processos /11/. Os objetivos desta tese são: a modelagem da interface entre os dispositivos gráficos de saída e o sistema gráfico de base, para minimizar a dependência interna de dispositivos gráficos de saída; e a aplicação dos resultados da modelagem na implementação de uma interface para um vídeo vetorial regenerativo.

Os capítulos seguintes estão organizados como segue.

No CAP.2, apresenta-se noções de computação gráfica,dando-se ênfase às características dos dispositivos gráficos convencionais e às estratégias que definem o comportamento e as características de um sistema gráfico de base.

No CAP.3, apresenta-se uma descrição resumida do GKS, abordando-se os principais conceitos que o norteiam.

No CAP.4, é discutido o modelo da interface de saída, dando-se ênfase a definição explicita, no sistema gráfico de base, das partes independente e dependente de dispositivos gráficos de saída. No final do capítulo apresenta-se um modelo funcional para a interface de saída.

No CAP.5, apresenta-se a aplicação do modelo da interface de saída na implementação de uma interface para um vídeo vetorial regenerativo.

Finalmente no CAP.6, apresenta-se um resumo dos resultados básicos obtidos por este trabalho e são feitas menções a trabalhos que dão sequência a esta tese.

# CAPÍTULO 2

# NOÇÕES DE COMPUTAÇÃO GRÁFICA

## 2.1 INTRODUÇÃO

Desde que a computação gráfica é uma área de pesquisa relativamente nova e com poucas publicações em nosso país, um número de têrmos e definições relativas à área precisam ser esclarecidos. Em particular, projeto assistido por computador ("CAD - computer aided design"), computação gráfica ("computer graphics"), computação gráfica passiva, computação gráfica interativa, e fabricação assistida por computador ("CAM - computer aided manufacturing") são frequentemente utilizados intercambiavelmente, ou de tal forma que causam confusão quanto a seus significados.

Destes termos, projeto assistido por computador é o mais geral, e pode ser definido como qualquer uso de um computador para ajudar o projeto de um subsistema ou de todo um sistema visando aumentar a produtividade do projetista, além de permitir um resultado melhor e de maior confiabilidade, pela possibilidade de se examinar várias alternativas em tempo hábil. Como subproduto importante, temos a possibilidade de armazenar em máquina todos os produtos do projeto. Este processo pode estabelecer uma conexão com a fabricação assistida por computador.

Computação gráfica é o conjunto de metodologias envolvendo a geração, representação, manipulação ou cálculo de objetos gráficos por computador, bem como a sua associação com informação de caráter não gráfico com eles relacionados / 1 /.Os objetos gráficos podem ser criados artificialmente usando caracteres alfanuméricos, simbolos especiais, linha e manchas coloridas ou sombreadas, mas podem também ser obtidas por fotografia.

Computação gráfica passiva é uma área da computação gráfica que prepara e apresenta informação gráfica ao usuário,mas não permite interação entre este e o sistema, conjunto de equipamentos e recursos reunidos para implantação da base física de um projeto.

Computação gráfica interativa é uma área da computação grá

fica que não apenas prepara e apresenta, mas também permite ao usuário interagir com o sistema com o propósito básico de manipular objetos gráficos.

Fabricação assistida por computador é qualquer uso de um computador na fabricação ou produção de subprodutos que auxiliem a execução de um processo.

A utilização da computação gráfica interativa em projeto e fabricação assistido por computador, fez com que esta área sofresse enorme desenvolvimento nas últimas décadas. Os dispositivos gráficos foram os que efetivamente possibilitaram avanços na utilização dos sistemas computacionais, através da otimização e ampliação da gama da funções ou serviços oferecidos por estes.

Entre estas funções e serviços podemos citar:

- Apresentação gráfica de resultados computacionais.
- Substituição do papel como meio de representação gráfica.
- Apresentação rápida de grande quantidade de informação.
- Visualização de objetos não existentes para um estudo de opções de projeto.
- Simulação de processos.
- Simulação de cenas reais.
- Criação de objetos artísticos.
- Entretenimento.

Muitas aplicações gráficas têm sido implantadas explorando estas funções ou serviços.

É interessante observar que o número de participantes em simpósios e atividades correlatas sobre computação gráfica tem crescido bastante ao longo dos últimos anos, o que demonstra a importância visivelmente crescente do assunto.

Neste capítulo deseja-se apresentar algumas noções de computação gráfica, principalmente as que auxiliem a compreensão da presente tese. Na próxima seção, são consideradas as características dos dispositivos gráficos mais comuns, bem como alguns dos critérios de classificação dos mesmos. Na seção 2.3,

são abordadas a importância e as características de uma linguagem gráfica padronizada, e as principais estratégias para a definição da mesma. Na última seção, são apresentadas as vantagens e desvantagens da normalização gráfica, e as principais características de dois trabalhos que se destacam nesta definição de normas.

## 2.2 HARDWARE GRÁFICO

Computação gráfica, durante as últimas décadas, tem sido fortemente influenciada pela tecnologia de dispositivos gráficos. Isto não é novidade, desde que o principal objetivo desta é o controle de dispositivos gráficos. Existe contudo, uma razão de maior relevância para esta influência: o uso de computação gráfica implica na comunicação gráfica homem - máquina, e portanto a qualidade da imagem apresentada e os mecanismos de interação são de extrema importância.

A maioria das técnicas de computação gráfica interativa desenvolvidas nas últimas décadas concentrou-se em dispositivos matriciais e vetoriais ("line-drawing"). Deste último destacam-se duas grandes classes: dispositivos regenerativos ("refresh") e os de armazenamento ("storage").

O vídeo vetorial regenerativo ("vector refresh display") é baseado no tubo de raios catódicos (CRT) comumente utilizado em televisores. Contudo, os métodos de geração de imagem são bastante diferentes. Televisores utilizam a técnica de varredura para gerar a figura, enquanto os vídeos gráficos vetoriais geram segmentos de reta de forma aleatória. Essencialmente o vídeo regenerativo necessita de três elementos adicionais ao tubo de raios catódicos; arquivo ("display file"), controlador ("display controller") e gerador gráfico (" display generator"). Estes três últimos são conhecidos como processador gráfico ("display processor"). A figura 2.1, apresenta o diagrama de blocos funcional de um vídeo regenerativo convencional. Um bloco ainda não mencionado, foi adicionado, consistindo do conjunto de dispositivos de entrada disponíveis ao usuário. O conjunto CRT e dispositivos de entrada é conhecido como console ("display console"). A discussão sobre os dispositivos de entrada é feita mais adiante.

De forma a compreendermos as vantagens e limitações do vídeo regenerativo é necessário compreendermos o propósito destes dispositivos. Desde que a luminescencia do fósforo no tubo de raios catódicos de vídeos regenerati

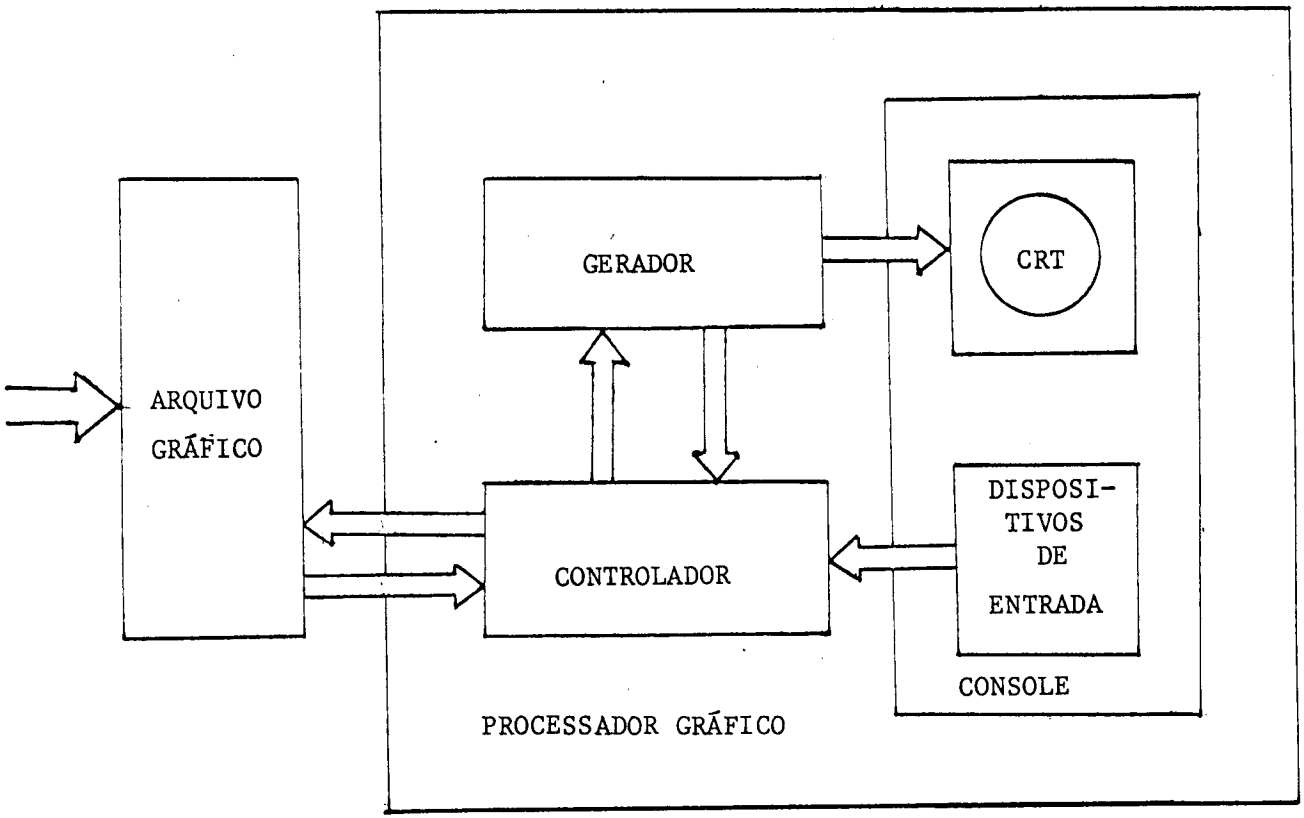



FIG. 2.1 - Diagrama de Blocos Funcional de um Vídeo Vetorial Regenerativo

vos decaí: rapidamente,isto é, tem baixa persistência, é necessário redesenhar toda a figura muitas vêzes por segundo. Isto é conhecido como taxa de regeneração e uma taxa muito baixa gerará um fenômeno denominado cintilação ("flicker"). O arquivo gráfico é um conjunto de instruções numa dada estrutura que permite ao processador gerar a imagem. Nela é armazenada a descrição completa da figura através de elementos primitivos como: vetores, pontos, áreas, etc. O controlador interpreta as instruções contidas no arquivo gráfico na taxa de regeneração e inicia sua execução controlando o gerador que por sua vez, produz sinais adequados para deflexão do feixe de elétrons no CRT, de modo a sensibilizá-lo para geração da imagem desejada. Imediatamente, uma limitação dos vídeos regenerativos aparece: a complexidade da figura é limitada pelo tamanho do arquivo gráfico e pela velocidade do controlador. Contudo, a baixa persistência da imagem pode ser usada vantajosamente para explorar modificações dinâmicas na imagem. Além disso, desde que cada elemento ou instrução necessária para desenhar toda a figura está no arquivo gráfico, qualquer elemento pode ser mudado, removido, ou ter uma característica adicional introduzida.

Os vídeos vetoriais da armazenamento ("storage tube display" ) essencialmente diferem dos vídeos regenerativos pela existência de uma estrutura armazenadora dentro do próprio tubo de raios catódicos. Isto implica, que explicitamente, não há necessidade de guardarmos informações gráficas em estruturas como o arquivo gráfico. O vídeo vetorial armazenador apresenta vantagens e desvantagens. Algumas destas vantagens são: o vídeo esta livre de cintilação, a resolução é boa, e o custo é comparativamente baixo. A principal desvantagem é que a figura não pode ser seletivamente removida como no caso do vídeo regenerativo.

Entre os dispositivos matriciais de acentuado desenvolvimento nos últimos anos, destacam-se os vídeos de varredura ("raster scan display"). Os vídeos de varredura utilizam um monitor de televisão convencional como console.No vídeo de varredura a figura é composta de uma série de pontos.O sinal elétrico básico utilizado para atuar no vídeo é um sinal analógico cuja modulação representa a intensidade de pontos individuais que compõem a figura. Ao utilizarmos um vídeo de varredura é necessário convertermos a informação de linha e caráter a uma forma compatível com o vídeo. Este processo é denominado conversão de varredura. O arquivo gráfico para o vídeo de varredura é normalmente arranjado numa matriz de intensidades conhecida como "frame buffer". Há três problemas basicos associados ao vídeo de varredura: o tamanho e o custo do "frame buffer",a conversão de varredura, e o efeito de quantização. Por exemplo, a questão da conver

são de varredura suscita o estudo da melhor forma de codificarmos a imagem. Estudos recentes indicam que a melhor forma de codificarmos está na utilização de segmentos de reta / 9 /. Isto implica em têrmos um arquivo gráfico similar ao usado em dispositivos vetoriais, podendo-se assim aproveitar-se o mesmo "software" utilizado pelos vídeos vetoriais para os de varredura.O problema desta abordagem esta em projetar-se um processador capaz de realizar esta conversão em tempo real. Um processador deste tipo pode ser visto em / 2 /. Ao considerarmos vídeos de varredura, as vantagens e desvantagens são similares às de vídeos vetoriais, com algumas características adicionais ;são geralmente mais lentos e difíceis de implementar remoção seletiva.

Não deixando de mencionar os dispositivos gráficos passivos mais utilizados, dos vetoriais,o mais comum é o "plotter", dos matriciais, é a impressora.

Existe uma variedade de dispositivos de entrada, e a qualidade da interação entre o usuário e o sistema dependerá das características funcionais dos dispositivos e os métodos de realimentação utilizados pelos mesmos,os quais devem ser adequados as necessidades de comunicação do usuário. Dispositivos de entrada podem ser agrupados de acôrdo com suas similaridades funcionais. Muitas classificações deste tipo tem sido propostas e a que tem encontrado maior aceitação / 3 /, divide-os nas seguintes classes:

PICK Dispositivos utilizados no processo na identificação de objetos.

BUTTON Dispositivos utilizados no processo como identificadores de função.

LOCATOR Dispositivos utilizados no processo para obtenção de informação de localização.

VALUATOR Dispositivos utilizados no processo para obtenção de um valor numérico.

KEYBOARD Dispositivos utilizados no processo de obtenção de uma sequência de caracteres.

Qualquer dispositivo físico pode operar em uma, algumas, ou mesmo em todas as classes descritas, pela adaptação de suas características ao tipo de informação solicitada pela classe em questão. Contudo, a funcionalidade desta utilização para o usuário é que determinará a sua adoção pelo sistema. Dentre os

dispositivos físicos da computação gráfica, possivelmente os mais conhecidos são o teclado,o"light pen",o"joystick", e o "tablet".

O teclado é um fornecedor de informação alfanumérica. Portanto, informação alfabética, numérica, ou de controle podem ser facilmente fornecidas.

O"light pen" é um dispositivo na forma de um lápis, contendo na sua extremidade uma célula fotoelétrica e circuíto associado que, quando posicionado sobre um segmento de reta ou outra área iluminada no CRT, a célula foto elétrica é ativada, causando uma interrupção. Normalmente, permite-se identificar a instrução apontada no arquivo gráfico.

O "joystick" é um dispositivo na forma de uma alavanca utilizada para controlar os movimentos de um cursor sobre a tela.

O "tablet" é uma superfície plana sobre a qual, através do uso de um estilete, é possivel digitalizar figuras.

Após termos vistos algumas características de tecnologia de dispositivos gráficos convencionais, convém observar-se alguns critérios de classificação dos mesmos. Algumas classificações já foram introduzidas, como, dispositivos gráficos passivos e interativos, vetoriais e matriciais. Cada critério fornece alguma perspectiva de compreensão dos dispositivos gráficos,dentro do algumas vêzes confuso, conjunto de dispositivos existentes. Uma classificação adicional parece-nos interessante acrescentar, a apresentada pelo SIGGRAPH / 4 /, que classifica os dispositivos gráficos segundo critérios de desempenho e capacidade de interação. Esta classificação distingue as seguintes categorias:

- Dispositivos passivos
- Dispositivos de baixo desempenho
- Dispositivos de alto desempenho

Vídeos vetoriais, de armazenamento e regenerativo, e vídeos de varredura normalmente são incluídos na segunda categoria.

## 2.3 SOFTWARE GRÁFICO

Muitas operações de entrada e saída em computador são realiza

das de uma forma padrão, utilizando linguagens de programação de alto nível. Por exemplo, linguagens como FORTRAN incluem facilidades para arquivos de entrada e saída. A possibilidade de expressar tais operações dentro de linguagens de alto nível torna a programação mais fácil e permite que os programas resultantes possam ser executados em computadores diferentes. Nós gostariamos que nosso programa de aplicação gráfica fosse igualmente portátil. Para isso, uma linguagem gráfica de propósitos gerais deveria ser definida, fornecendo a seu usuário a possibilidade de descrever, gerar e manipular figuras. Para descrever e gerar uma figura, a linguagem precisa ter comandos pelos quais primitivos gráficos são especificados com certos atributos determinando a sua aparência no console. A manipulação requer o emprego de transformações geométricas e de "windowing", e a possibilidade de rearranjar, misturar ou remover elementos da figura.

Para facilitar a portabilidade dos programas gráficos, é desejável que os recursos da linguagem descrita estejam disponíveis através de um pacote gráfico. O pacote gráfico é um conjunto de subrotinas que permitem o acesso aos recursos gráficos de entrada e saída. Um pacote gráfico simplifica a tarefa do programador e torna possível escrever programas portáteis que possam rodar em diferentes computadores com diferentes dispositivos gráficos.

O projeto de um pacote gráfico de propósito gerais é um dos problemas centrais da computação gráfica. Este tipo de pacote precisa prover uma quantidade de funções, envolvendo conhecimento das necessidades das várias aplicações da computação gráfica.

Apesar da quantidade de conhecimento existente antes de 1976, uma metodologia para o projeto de padronização gráfica ainda não existia. Entre 1976 e 1977, uma grande passo foi dado objetivando a obtenção de uma metodologia para a padronização gráfica. Desses esforços, originou-se uma diferença conceitual que, mesmo artificialmente, determinou que as transformações em aplicações gráficas são utilizadas para dois propósitos diferentes; visualização ("viewing") e modelamento ("modelling"). Esta conceituação foi um dos primeiros resultados importantes para o estabelecimento de uma estratégia para a padronização gráfica / 5 /. O que significa esta conceituação e as consequências que esta traz para a padronização gráfica são discutidas a seguir.

Em programas de aplicação gráfica, várias transformações são utilizadas em diferentes fases do processamento gráfico. O têrmo transformação é

usado aqui com um significado mais amplo do que aquele normalmente utilizado.Nós não estamos aqui nos referindo apenas a transformações geométricas, mas também a operações tais como, remoção de superficies escondidas ou produção de objetos sólidos sombreados.

Em muitas aplicações de computação gráfica tridimensional, as técnicas de projeção convencionais podem não satisfazer as necessidades do usuário, podendo este utilizar outras técnicas de transformação. Nos casos em que as transformações convencionais não atendem as necessidades do usuário, um conjunto de transformações especiais deve ser utilizado para produzir uma figura em coordenadas reais e, sómente então, as formas de transformações convencionais são utilizadas para criar a imagem na superfície de visualização.

Transformações de visualização, como "window/viewport", são entendidas como distintas das transformações de modelamento, tais como as utilizadas para compor objetos sólidos em estruturas complexas.

Em outras palavras, as transformações de visualização aplicadas a um objeto dizem respeito à representação deste em uma tela, ou seja, este objeto como um observador o vê. As transformações de modelamento dizem respeito à construção, composição ou montagem de um objeto, sem qualquer consideração de visualização. As transformações de visualização farão parte do que será conhecido como sistema gráfico de base. Espera-se que este sistema seja independente da aplicação e, o quanto possível, independente de dispositivos.

Um modelo funcional de todo sistema, voltado para o programador é mostrado na figura 2.2. Um conjunto especial de funções de modelamento, na forma de um pacote separado de subrotinas é utilizado para decompor a estrutura de dados da aplicação em uma definição da figura em coordenadas reais. Um pacote gráfico de propósitos gerais é então utilizado para aplicar-se as transformações de visualização a esta figura, depositando esta imagem no arquivo gráfico.

As funções de modelamento variarão de uma aplicação para outra; em aplicações mais simples podem ser quase inexistentes, enquanto em outras, podem ter um grande número de funções.

O aspecto mais importante desta formalização do modelo do programador é a sua modularidade.As funções gráficas que dizem respeito a geração de primitivas: funções primitivas, funções para manipulação de segmentos, e funções

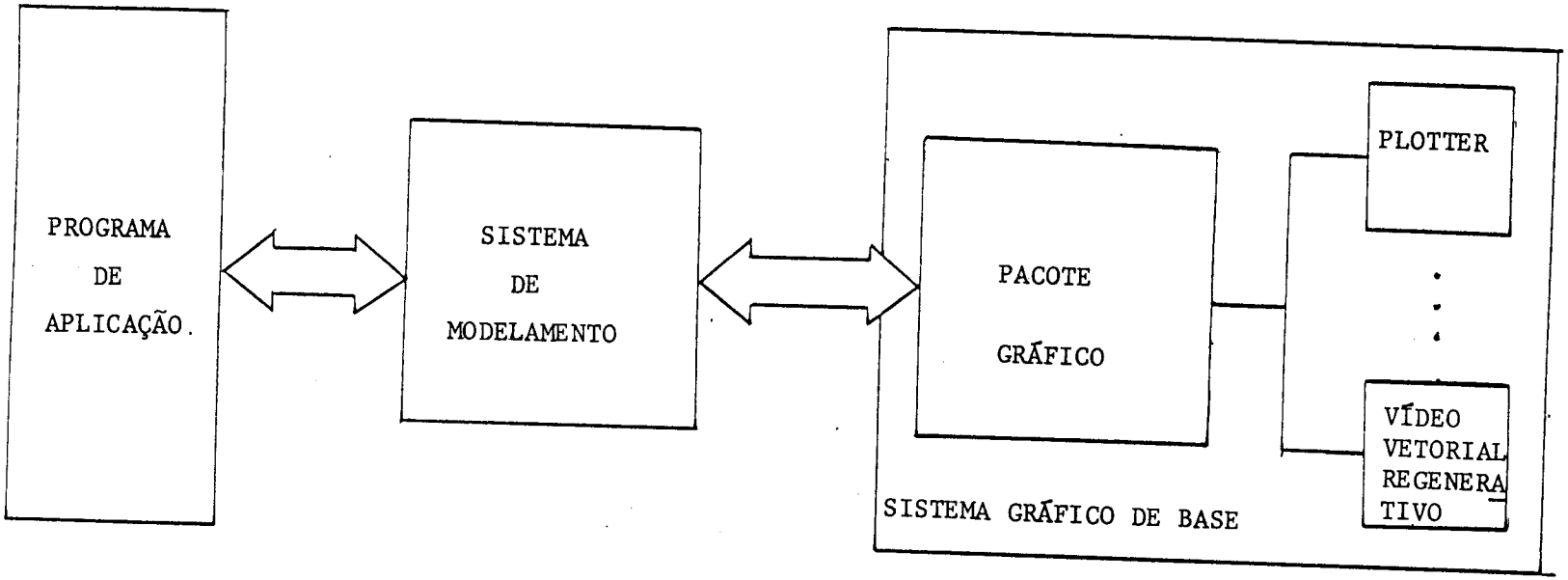



FIG. 2.2 - Modelo do Programador

para transformações de visualização são agrupadas como um módulo, o sistema gráfico de base. As funções de modelamento são separadas em outro módulo, o sistema de modelamento. Diferentes sistemas de modelamento podem ser desenvolvidos para diferentes aplicações, mas deve existir um único sistema gráfico de base para manusear toda a geração de figuras. De modo geral, a abordagem modular permite o desenvolvimento de pacotes especializados, sem sobrecarregar o sistema gráfico de base com funções complexas e/ou raramente utilizadas.

Esta abordagem modular, pode mesmo, ser utilizada para a própria implementação do sistema gráfico de base, permitindo-se separar aquelas funções que podem ser intituladas independentes de dispositivos, daquelas que não o podem, ou seja, são dependentes de dispositivos. Neste conjunto dependente de dispositivos, destacam-se os módulos de entrada e saída.

## 2.4 NORMALIZAÇÃO GRÁFICA

No processo de normalização (padronização) gráfica, discutiu se na seção anterior a divisão das necessidades de aplicações gráficas em dois sistemas distintos. Algumas considerações adicionais sobre as vantagens e desvantagens da normalização, que esclareçam os motivos e objetivos desta, bem como mostrem onde concentrar-se-ão os esforços para a especificação de um sistema gráfico de base, são discutidos a seguir.

Usualmente, três vantagens são associadas a normalização gráfica: independência de dispositivos, portabilidade do programa, e portabilidade do programador / 4 /.

Por independência de dispositivos entende-se a possibilidade de utilizar-se a mesma aplicação gráfica com qualquer dispositivo gráfico disponível em uma dada configuração.

Portabilidade do programa, possivelmente a maior vantagem da normalização, é a possibilidade de transportarmos uma aplicação gráfica de uma instalação para outra com um mínimo de modificações. De forma pragmática, o programa pode ser considerado portátil se o custo para adaptar um programa é menor do que o de reescrevê-lo.

A última vantagem, a portabilidade do programador, é a pos

sibilidade de um programador de aplicação mudar de um sistema para o outro com um retreinamento mínimo.

A principal desvantagem é a existência de uma variada gama de dispositivos gráficos de arquitetura e desempenho diferentes, o que pode nos levar à subutilização das potencialidades de alguns tipos de dispositivos gráficos.

A figura 2.3, mostra o modelo da figura 2.2., reconfigurado para mostrar o tipo de portabilidade de programa mais comumente encontrada; o programa de aplicação e o sistema de modelamento precisam ser movidos juntos, quando há mudança do programa de aplicação de uma instalação para outra.

Os trabalhos que estão sendo realizados por grupos internacionais para a especificação de um sistema gráfico de base situam-se na definição de características da interface padronizada apresentada na figura 2.3.

Neste esforço para a definição de normas, dois trabalhos destacaram-se em sua especificação: o primeiro, CORE SYSTEM, foi apresentado por um subgrupo do GSPC ("graphics standards planning committee"), conhecido como SIGGRAPH ("special interest group for computer graphics") em 1977, e o segundo, GKS (" grafical kernel system"), foi apresentado pelo NI ("normung institute") também em 1977; em ambos um pacote gráfico é proposto.

O CORE aceita a descrição da geometria de objetos no espaço bidimensional ou tridimensional como combinação de linhas, sequências de linhas, marcadores, texto, etc. A posição das primitivas de saída pode ser especificada em coordenadas absolutas ou relativas. Atributos afetam a aparência de primitivas de saída. Uma vez que a primitiva é criada, seus atributos podem ser mudados sòmente pela remoção da primitiva,utilizando a entidade segmento descrita a seguir, e então, reespecificando-a.

Todas as primitivas de saída são colocadas em segmentos, que formam a unidade básica de manipulação. Figuras podem ser mostradas em um ou mais dispositivos gráficos.

A classe de dispositivos de entrada suportados pelos dois sistemas são, com pequenas variações, aquelas apresentadas na seção 2.2.

O GKS aceita a descrição da geometria de objetos apenas no espaço bidimensional, deixando as operações de projeção para o sistema de modelamento.

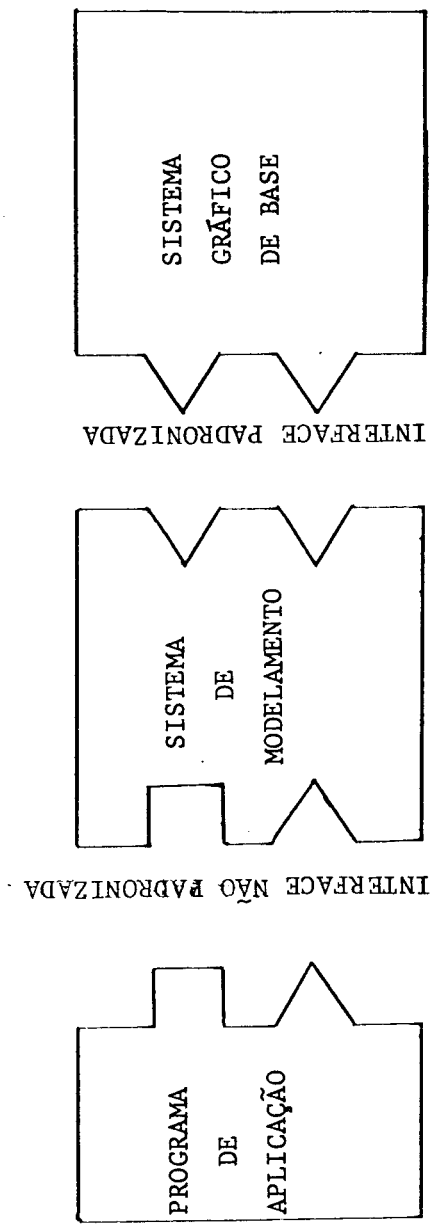
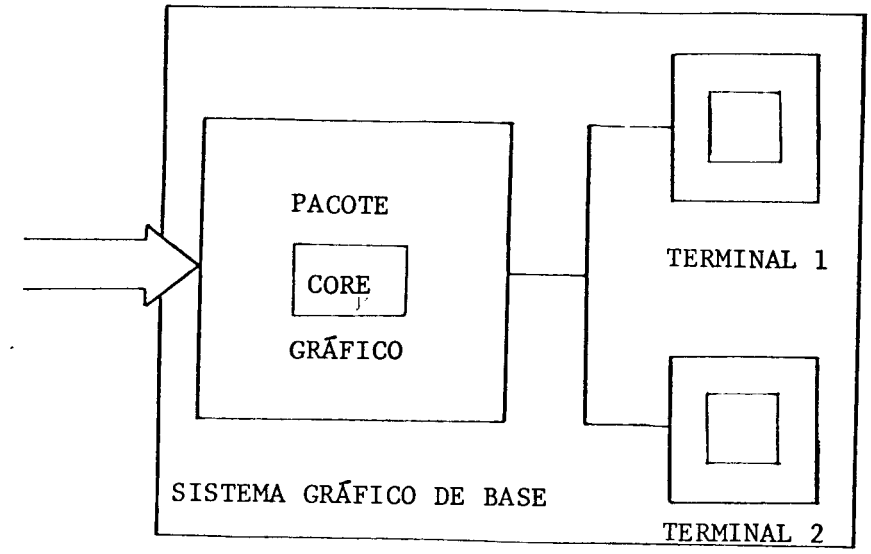



FIG. 2.3 - Tipo de Portabilidade Comumente Encontrada

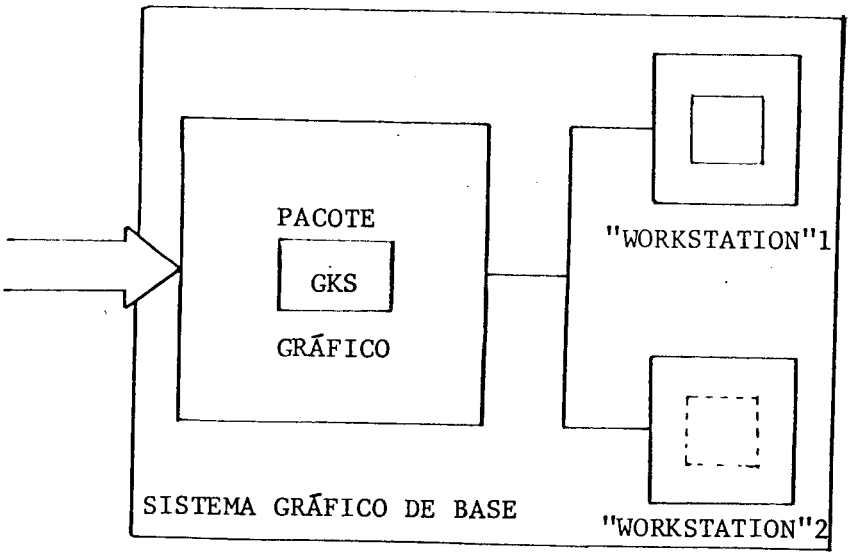
Seus objetos são descritos como sequências de linhas, sequências de marcadores, texto, etc. Contudo, a posição das primitivas de saída é especificada apenas em coordenadas absolutas e depois da criação de uma primitiva, seus atributos podem ser modificados sem necessidade de remover-se a primitiva.

O conceito que o distingue e possibilita um melhor aproveitamento das capacidades de dispositivos gráficos, é o de "workstation". Uma "workstation" representa uma coleção de dispositivos gráficos que são operados de uma forma coordenada e é tratada pelo GKS como uma única unidade lógica.

Uma das vantagens que provem do conceito de "workstation" é ilustrada na figura 2.4. Supondo que em ambos os sistemas os terminais gráficos estão prontos para operar , no CORE, ao criarmos um segmento que descreve um quadrado, alguns atributos são estáticos. No caso, se a característica de linha tipo sólido for anexada à figura, esta aparecerá com o mesmo tipo de linha em todos os terminais em operação como mostrado na figura 2.4a. No GKS, a aparência da primitiva é definida em dois estágios. No primeiro, um símbolo é associado a um conjunto de primitivas, no segundo, os atributos do símbolo são mapeados de acôrdo com a capacidade da "workstation",determinando a aparência das primitivas no terminal. A associação do símbolo a suas características é obtida em uma tabela denominada tabela de estado da "workstation". Assim, ao associar-se o símbolo 1 ao segmento em questão, para a "workstation" 1, isto pode significar linha do tipo sólido, enquanto para a "workstation" 2, linha do tipo tracejado, como mostrado na figura 2.4b. A melhor utilização de dispositivos gráficos de características diversas, pode justificar a adoção do GKS como núcleo para a implementação de um sistema gráfico de base.




(a)




(b)

FIG. 2.4 - Exemplo Comparativo CORE/GKS

# CAPÍTULO 3

# DESCRIÇÃO SUCINTA DO GKS

## 3.1 PROPÓSITOS E CARACTERÍSTICAS :

Esta normatização, GKS, especifica um conjunto de funções para o processamento de dados gráficos, de forma a torná-las independentes de dispositivos gráficos, linguagens de programação ou aplicações / 6 /.

O "graphical kernel system" aceita a descrição de objetos no espaço bidimensional, cada objeto é construído a partir de primitivas de saída. Existem três primitivas vetoriais, uma de texto, e duas de "raster".

O GKS é baseado no conceito de "workstation" que distingue-se de sua realização física, ao qual denominamos terminal gráfico. Uma " workstation"é uma coleção de dispositivos gráficos, tratados como uma unidade lógica. Muitas funções que em pacotes gráficos de propósitos gerais eram comuns a todos os terminais gráficos, são tratados pelo GKS localmente. Por exemplo,a transformação de coordenadas é um processo de dois estágios; no primeiro, a mesma transformação é aplicada a toda primitiva;no segundo, uma em cada "workstation". Além disso, o controle de tabelas de "pen" e "text" na "workstation" permite o controle da aparência das primitivas localmente.

O conceito de segmento fornece meios de estruturarmos uma figura em subpartes. Segmentos podem ser criados, removidos, ter os seus atributos mudados dinamicamente, e podem ser transformados. Um armazenador de segmento independente de dispositivos ("device independent segment storage") serve para inserir segmentos dentro de outros.

O GKS define somente o núcleo de um sistema gráfico de base. Para integrá-lo a uma linguagem, precisa-se conectá-lo a uma camada dependente da linguagem.

O modelo de camadas mostrado na figura 3.1, ilustra a aplicação do GKS em um sistema gráfico em geral; o programa do usuário usará a camada de modelamento, a camada dependente da linguagem, e os recursos do sistema o

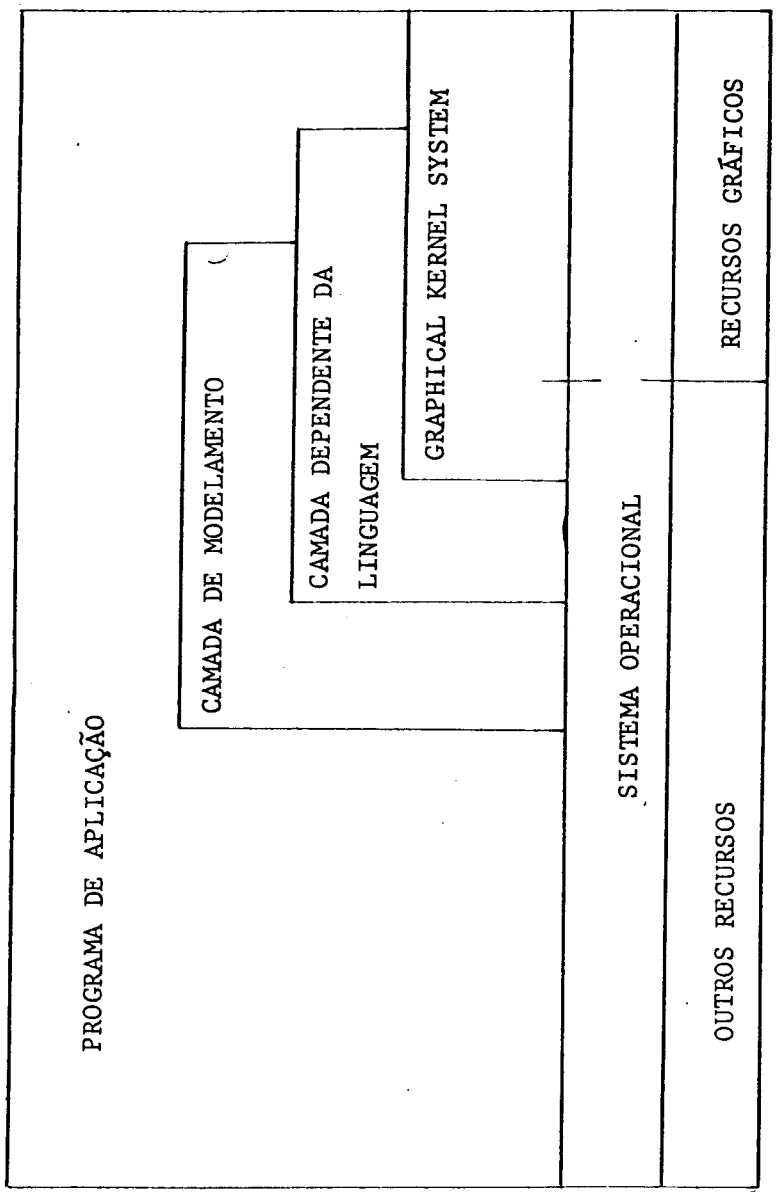



FIG. 3.1 - Modelo de Camadas para o GKS

peracional. Os recursos gráficos só podem ser acessados via GKS.

## 3.2 MODELO DO USUÁRIO

No projeto de qualquer sistema computacional, é necessário fornecer ao usuário do sistema um modelo simples e coerente. Sem este o usuário terá dificuldade em operar o sistema, prejudicando o desempenho do mesmo. Este princípio aplica-se também ao projeto de um sistema gráfico.

O conhecimento básico que o programador deve possuir inclui:

- Separação das funções de entrada e saída
- O conceito de três sistemas de coordenadas:
  - Sistema de coordenadas reais
  - Sistema de coordenadas normalizadas
  - Sistema de coordenadas do dispositivo
- O conceito de segmento
- O conceito de "workstation"

A figura 3.2, mostra o fluxo de dados de um programa gráfico para uma "workstation" interativa. O programa de aplicação cria a definição dos objetos a serem mostrados. Esta definição, após a primeira operação de transformação, pode ser armazenada em uma estrutura de dados denominada armazenador de segmento ("segment storage") e/ou encaminhada a todas as "workstations" ativas. De qualquer forma, a definição da figura é inicialmente apresentada ao sistema em coordenadas reais. Ela então passa por uma transformação geral comum a todas as primitivas, para gerar a representação da figura em coordenadas normalizadas. Esta representação pode ser vista como uma figura em uma tela virtual independente de qualquer dispositivo gráfico. A partir daí, para cada "workstation" ativa e transformação associada, é gerada a representação da figura em coordenadas do dispositivo.Entradas pelo console podem fazer com que o programa de aplicação calcule novos dados e/ou modifique a imagem apresentada na tela.

## 3.3 PRIMITIVAS DE SAÍDA

Cada figura que o programador descreve consiste de um ou mais objetos. Cada objeto gráfico é descrito por primitivas de saída . As primi

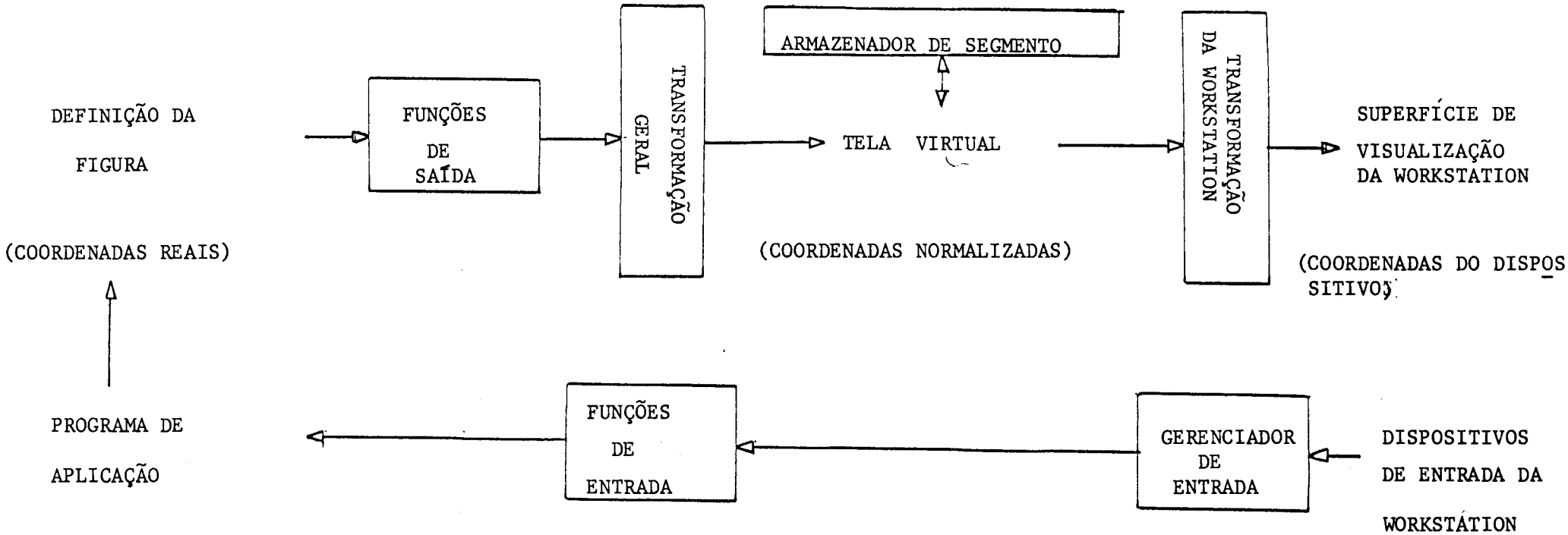



FIG 3.2 - Fluxo de Dados Para Uma "Workstation" Interativa.

tivas de saída geradas pelo programa de aplicação são encaminhadas a todas as "workstations" ativas. O GKS fornece as seguintes primitivas:

| | |
|---|---|
| POLYLINE | GKS gera uma sequência de segmentos de reta definida por uma sequência de pontos. |
| POLYMARKER | GKS gera uma sequência de símbolos centrados em posições dadas. |
| DRAW | Primitiva geral para atender a solicitação de primitivas especiais tais como círculo, arco de círculo, etc. |
| TEXT | GKS gera uma sequência de caracteres a partir de uma posição dada. |
| FILL AREA | Um poligono é preenchido com uma cor uniforme. |
| PIXEL ARRAY | Uma matriz de "pixels" com cores individuais é desenhada. |

Cada primitiva está associada com atributos que, de acordo com suas características, determinam sua aparência e/ou as identificam. Os atributos de primitivas de saídas são tratados na seção 3.8.1.

## 3.4. PRIMITIVAS DE ENTRADA

O GKS adotou o sistema de classificação de dispositivos de entrada, essencialmente idêntico ao mostrado na seção 2.2, substituindo-se as palavras "Button" e "Keyboard", por " choice" e"string", respectivamente. As entradas gráficas podem ser obtidas de qualquer "workstation" aberta. As primitivas de entrada são então classificadas em cinco classes, que especificam o tipo de primitiva de entrada, e em três modos de operação. As cinco classes de entrada são:

| | |
|---|---|
| LOCATOR | Fornece uma posição em coordenadas reais |
| VALUATOR | Fornece um número real |
| CHOICE | Fornece uma alternativa |
| PICK | Fornece o nome do segmento e o identificador de "pick" |
| STRING | Fornece uma sequência de caracteres |

Todas as cinco classes de entrada podem ser obtidas de uma

"workstation" pelo uso de três mecanismos diferentes. Os três modos de entrada são:

REQUEST — GKS solicita e lê uma primitiva de entrada de uma dada classe na"workstation". Equivalente ao READ ou ACCEPT em FORTRAN.

SAMPLE — GKS inspeciona o dado corrente em um dispositivo de entrada e retorna esse dado sem a necessidade de espera de uma ação do operador.

EVENT — GKS constroi uma fila de entrada, na qual primitivas de entrada de várias fontes são colocadas na sequência de geração temporal. Esta fila pode ser inspecionada pelo programa de aplicação.

## 3.5 SISTEMA DE COORDENADAS

O GKS considera três sistemas de coordenadas:

- Sistema de coordenadas reais
- Sistema de coordenadas normalizadas
- Sistema de coordenadas do dispositivo

Dois passos de transformação são aplicados às primitivas de saída encaminhadas pelo programa de aplicação: no primeiro, uma transformação de "window/viewport", possivelmente seguida por uma operação de "clipping", é aplicada a todas as primitivas realizando o mapeamento do espaço de coordenadas reais para o espaço de coordenadas normalizadas; no segundo, uma transformação de" window/viewport" específica de cada "workstation", com a respectiva operação de "clipping", faz o mapeamento do espaço de coordenadas normalizadas para o espaço de coordenadas do dispositivo.

Isto permite a exibição de várias vistas da mesma figura em "workstations" distintas, com diferentes escalas. A figura 3.3, mostra um exemplo da aplicação destas transformações.

## 3.6 O CONCEITO DE SEGMENTO

As primitivas de saída que descrevem um objeto podem ser ge

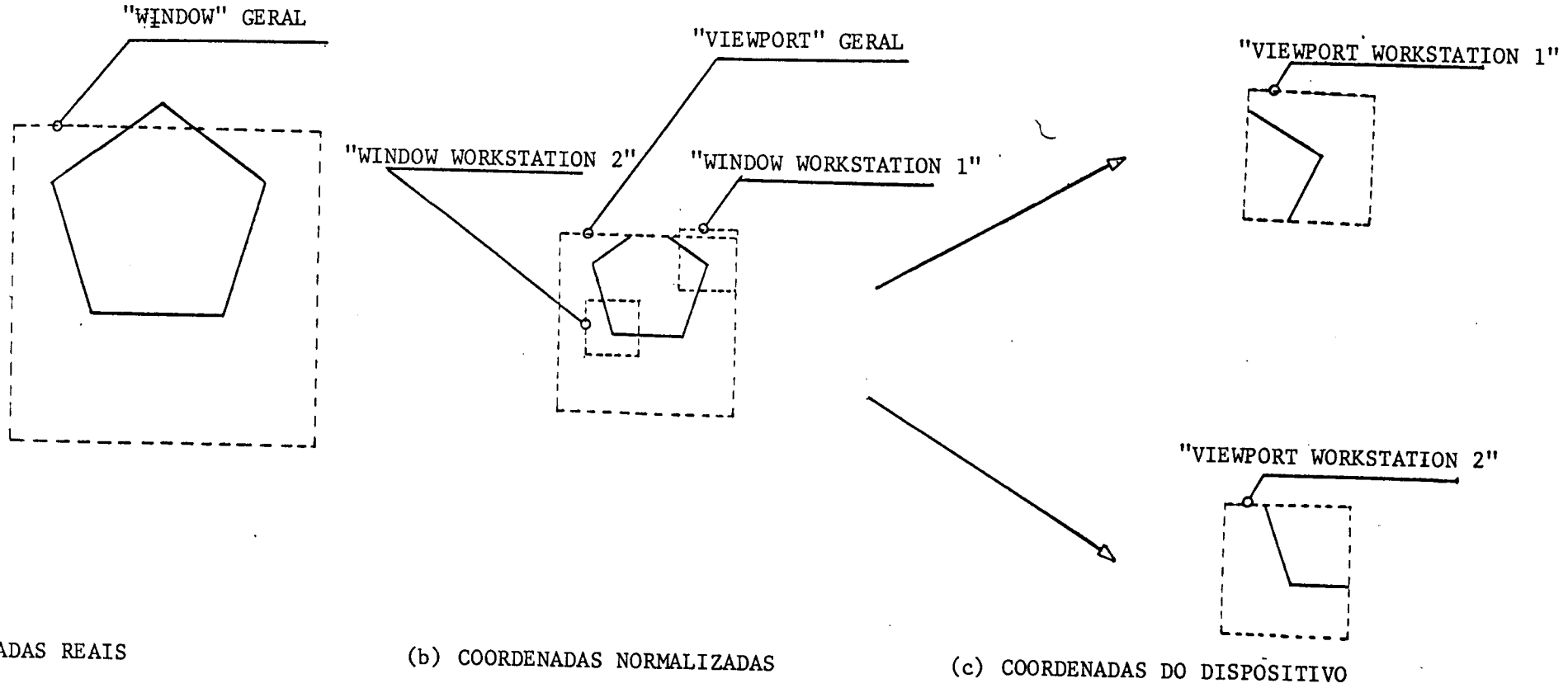



FIG:3.3 - Efeitos das Transformações de Imagem

radas fora de um segmento ou podem ser colocadas dentro de um segmento identificado por um nome. O programa de aplicação não tem acesso às primitivas fora do segmento depois de sua geração. Para utilizar um segmento, o programa de aplicação abre-o com um nome, chama as primitivas e os atributos que descrevem o objeto, e fecha-o. O segmento é a unidade básica de manipulação e pode ter o nome e a prioridade mudada,ser removido, transformado ou inserido, ser feito visível ou invisível, piscável ou não piscável, e detetável ou não detetável. Toda primitiva dentro de um segmento tem um identificador de "pick" associado, o qual estabelece um segundo nível de identificação de primitivas. Ao identificador de "pick" é dado o valor "default" zero quando o segmento é aberto. O valor -1 para o identificador de "pick" torna as suas primitivas associadas não detetáveis.

Quando o segmento é fechado, as suas primitivas não podem ser modificadas ou removidas. Contudo, transformações geométricas, mudança dos atributos do segmento e na tabela de "pen" e "text" em uma"workstation" específica são possíveis. Os atributos de segmento são discutidos na secção 3.8.

Segmentos podem sofrer transformações geométricas de escala,rotação, translação, ou qualquer combinação destas. Segmentos armazenados no armazenador de segmento podem sofrer transformações, e serem inseridos em um segmento aberto.

## 3.7 O CONCEITO DE WORKSTATION

Um dos princípios básicos do GKS é o conceito de "workstation". Todo tipo de "workstation" presente numa dada implementação possui uma tabela que descreve suas capacidades e características,denominada tabela de descrição da "workstation".Para toda "workstation" aberta seu estado é guardado em uma tabela, denominada tabela de estado da "workstation". Uma"workstation" totalmente equipada inclui:

- Uma superfície de visualização
- Permite diversos tipos de linha, tamanho de caracteres,etc.
- Tem um ou mais dispositivos de entrada para cada classe de entrada
- Permite entradas no modo "request","sample" e "event"

Numa instalação a "workstation" pode ou não estar equipada com

todas estas capacidades. O programa de aplicação pode utilizar funções de inquisição ("inquire") para solicitar quais capacidades estão disponíveis. Caso o programa de aplicação solicite funções que a "workstation" não provê um erro padrão está definido.

As "workstation" se enquadram em três categorias:

- "Workstation"de saída
- "Workstation"de entrada
- "Workstation" interativa

As funções de controle de "workstation" são também utilizadas para o armazenamento de informação gráfica, como é o caso do armazenador de segmento.

### 3.7.1 Utilização

As "workstations" são referenciadas pelo programa de aplicação através de um identificador de "workstation". Quando o programa de aplicação deseja trabalhar com uma "workstation", ele precisa inicialmente abri-la ("open workstation"). A ação de abrir associa a "workstation" ao terminal gráfico correspondente e dá ao programa de aplicação acesso a todas as suas capacidades, exceto a saída de primitivas gráficas. Para saída de primitivas a "workstation" precisa ser explicitamente ativada ("active workstation").

### 3.7.2 Adiamento de Mudança

A superficie de visualização da "workstation", sempre que possível, deve refletir o estado atual da figura como definido pelo programa de aplicação. Contudo, para utilizar-se eficientemente as capacidades das "workstations", é desejável permitir à "workstation" o adiamento de algumas ações solicitadas pelo programa de aplicação. Inclusas nestas, estão todas as funções que levem a uma regeneração implícita da "workstation". Funções que realizam regeneração implícita apagam todas as primitivas fora de segmento na superfície de visualização.

A função de adiamento ("set deferral state") permite escolher um estado da figura que leve em conta as capacidades da "workstation" e as necessidades do programa de aplicação. O GKS especifica os seguintes estados de adiamento:

ESTADO 1: O efeito visual de cada função tem de se tornar visível tão rápido quanto possível.

ESTADO 2: O efeito visual de cada função tem de se tornar visível antes da próxima interação.

ESTADO 3: O efeito visual de cada função deve ser adiado.

ESTADO 4: Idêntico ao Estado 1, mas com regeneração implícita suprimida.

ESTADO 5: Idêntico ao Estado 2, mas com regeneração implícita suprimida.

ESTADO 6: Idêntico ao Estado 3, mas com regeneração implícita suprimida.

## 3.8 ATRIBUTOS

Nesta seção abordam-se os três conjuntos de atributos; das primitivas de saída, de segmento, e da "workstation".

### 3.8.1 Atributos das Primitivas de Saída

Num programa de aplicação, quando é encontrada uma referência a um atributo de primitivas, as primitivas que forem definidas depois deste, e que sejam afetadas pelo mesmo, serão a ele associadas.

Isto faz parte de uma filosofia de atribuição de características em dois estágios: no primeiro, um símbolo é associado a um conjunto de primitivas; no segundo, as características que o símbolo descreve são atribuídas às primitivas de saída, determinando sua aparência na superficie de visualização.

Os atributos podem ser classificados em estáticos e dinâmicos. No primeiro caso, um símbolo com certas características é associado a um conjunto de primitivas e, no momento da geração na "workstation" estas características são mapeadas nas primitivas e não podem mais ser mudadas, a não ser pela remo

ção e redefinição da primitiva. Nesta classe temos os atributos: espaçamento e tamanho de texto, tamanho de "marker", e identificador de "pick".

Os dinâmicos são aqueles que, mesmo após seu mapeamento na "workstation", permitem modificação de suas características, sem necessidade de redefinição das primitivas. Nesta classe incluem-se os atributos; número "pen" e "text".

Tamanho e espaçamento de texto referem-se à especificação de caracteres de um texto e aplicam-se somente à primitiva TEXT.

O tamanho de"marker" especifica o tamanho dos marcadores e aplica-se somente à primitiva POLYMARKER.

O atributo identificador de "pick" identifica um conjunto de primitivas em um segmento quando apontado por um dispositivo de entrada da classe "pick". Aplica-se a todas as primitivas de saída.

O número "text" é um índice que faz referência a uma tabela existente em cada "workstation"; esta tabela conterá informações como tipo de caracter ("font") e qualidade do texto. Aplica-se somente à primitiva do tipo TEXT.

O número "pen" é um índice que faz referência a uma tabela existente em cada "workstation"; esta tabela conterá informações como tipo de linha, largura de linha e intensidade/cor. As características de "pen" podem ser aplicadas a diversas primitivas; em algumas primitivas apenas algumas destas são aplicáveis. A tabela 3.1, apresenta as características de "pen" aplicáveis às diversas primitivas.

| | Tipo de Linha | Largura de Linha | Cor/Intensidade |
|---|---|---|---|
| POLYLINE | X | X | X |
| POLIMARKER | o | o | X |
| DRAW | o | o | o |
| TEXT | o | o | X |
| FILL AREA | - | - | X |
| FILL ARRAY | - | - | X |

X = válido  o = pode ser válido  - = não é válido

Tabela 3.1 - Primitivas de saída e as características de "pen".

### 3.8.2 Atributos de Segmento

Os atributos de segmento são únicos para cada segmento e portanto, são os mesmos para "workstations" diferentes. Os atributos são dinâmicos, podendo mudar a qualquer momento. Atributos "default" são atribuidos a um segmento quando o mesmo é aberto. Os atributos de segmento são:

| | |
|---|---|
| VISIBILIDADE | um segmento é mostrado ou não |
| DETETABILIDADE | um segmento pode ser selecionado por um dispositivo da classe pick ou não. |
| PISCAMENTO | um segmento é piscável ou não |
| PRIORIDADE | se partes de segmentos se sobrepoem, o de maior prioridade será selecionado |
| TRANSFORMAÇÃO | veja seção 3.6 |

### 3.8.3 Atributos da "Workstation"

O GKS especifica ainda um conjunto de atributos de uma "workstation". Estes atributos controlam a aparência das primitivas de saída em uma"workstation"específica. Estes atributos são representação de "pen" e "text", estado de adiamento, e transformação na "workstation".

A representação de um número "pen" ou "text" emuma"workstation" é encontrada na tabela de estado da "workstation". Algumas definições iniciais estão presentes na tabela de descrição da "workstation" e são usadas como "default". O programa de aplicação pode utilizar uma definição "default" ou pode especificar uma representação mais conveniente para o mesmo.

Os atributos da"workstation" podem ser diferentes em diversas "workstations" e podem ser alterados. Modificações dinâmicas aceitas,contidas na tabela de descrição da "workstation", indicam que mudanças podem ser realizadas i

mediatamente, podem ser adiadas, ou não são aceitas.

## 3.9 O CONCEITO DE NÍVEIS E ESTADOS

O GKS pode ser usado para uma grande variedade de aplicações, desde saída passiva até movimentação dinâmica e interação em tempo real. Contudo, o sistema pode se mostrar insatisfatório sob certas condições, ou porque lhe faltam recursos, ou porque os recursos disponíveis degradam o seu desempenho. Por exemplo, o usuário de um "plotter" pode questionar a necessidade dos recursos de processamento para se manter uma estrutura de segmentação. É comum encontrar - se uma variação suficientemente grande nas necessidades do programador de aplicação e nas características funcionais dos dispositivos, que justificam níveis ascendentes e compatíveis de implementações do GKS.

As capacidades funcionais do GKS podem ser agrupados nos seguintes grupos: básico, de saída, de entrada e de segmentação. O grupo básico é essencial para qualquer implementação. Os grupos de saída, de entrada, e de segmentação podem ser subdivididos, baseados em níveis de complexidade. Seis níveis ascendentes de implementação para o GKS foram definidos.

O GKS define cinco estados operacionais; funções do programa de aplicação somente são permitidas em certos estados. Estes estados operacionais são:

0 = GKS fechado;
1 = GKS aberto;
2 = Pelo menos uma workstation aberta;
3 = Pelo menos uma workstation ativa;
4 = Segmento aberto;

Entre a chamada de duas funções pelo programa de aplicação, o estado completo do GKS é definido por um conjunto de variáveis de estado com valores específicos. Estas variáveis de estado caracterizam-se pelo fato de permitirem uma descrição de todos os efeitos das funções. As tabelas que descrevem o estado completo do GKS são:

- Tabela de estado do GKS
- Tabela de estado de segmento
- Tabela de estado da"workstation"
- Tabela de estado de erro do GKS

# CAPÍTULO 4

# MODELO DA INTERFACE DE SAÍDA

## 4.1 SISTEMA GRÁFICO

A implementação de um sistema gráfico de base, tendo como núcleo o GKS, será facilitada pela divisão do sistema num conjunto de módulos autônomos. Uma descrição preliminar para implementação de um sistema gráfico / 7 /,especifica um modelo para a sua implementação. Este modelo é mostrado na figura 4.1 e é constituido de cinco módulos básicos:

- Interface do usuário
- Interface para um banco de dados
- Interface de saída
- Interface de entrada
- Núcleo gráfico

A interface do usuário terá a função de adaptar o sistema gráfico de base a uma aplicação específica; corresponde ao sistema de modelamento apresentado na seção 2.3.

A interface para um banco de dados liberará o núcleo gráfico da tarefa de gerenciamento das estruturas de dados das tabelas do GKS, como também do armazenamento de programas gráficos.

A interface de saída fará a conexão entre o núcleo gráfico e os vários periféricos de saída. Esta interface adequará o comportamento do periférico às capacidades que o núcleo prevê para o mesmo. A definição das necessidades e características desta interface, é o assunto da presente tese.

A interface de entrada fará a conexão entre o núcleo gráfico e os vários dispositivos de entrada. O trabalho de R.P. Martins / 8 /, que esta sendo desenvolvido em paralelo a esta tese, define as necessidades e características da interface de entrada.

O núcleo gráfico é o centro do sistema, um conjunto de rotinas

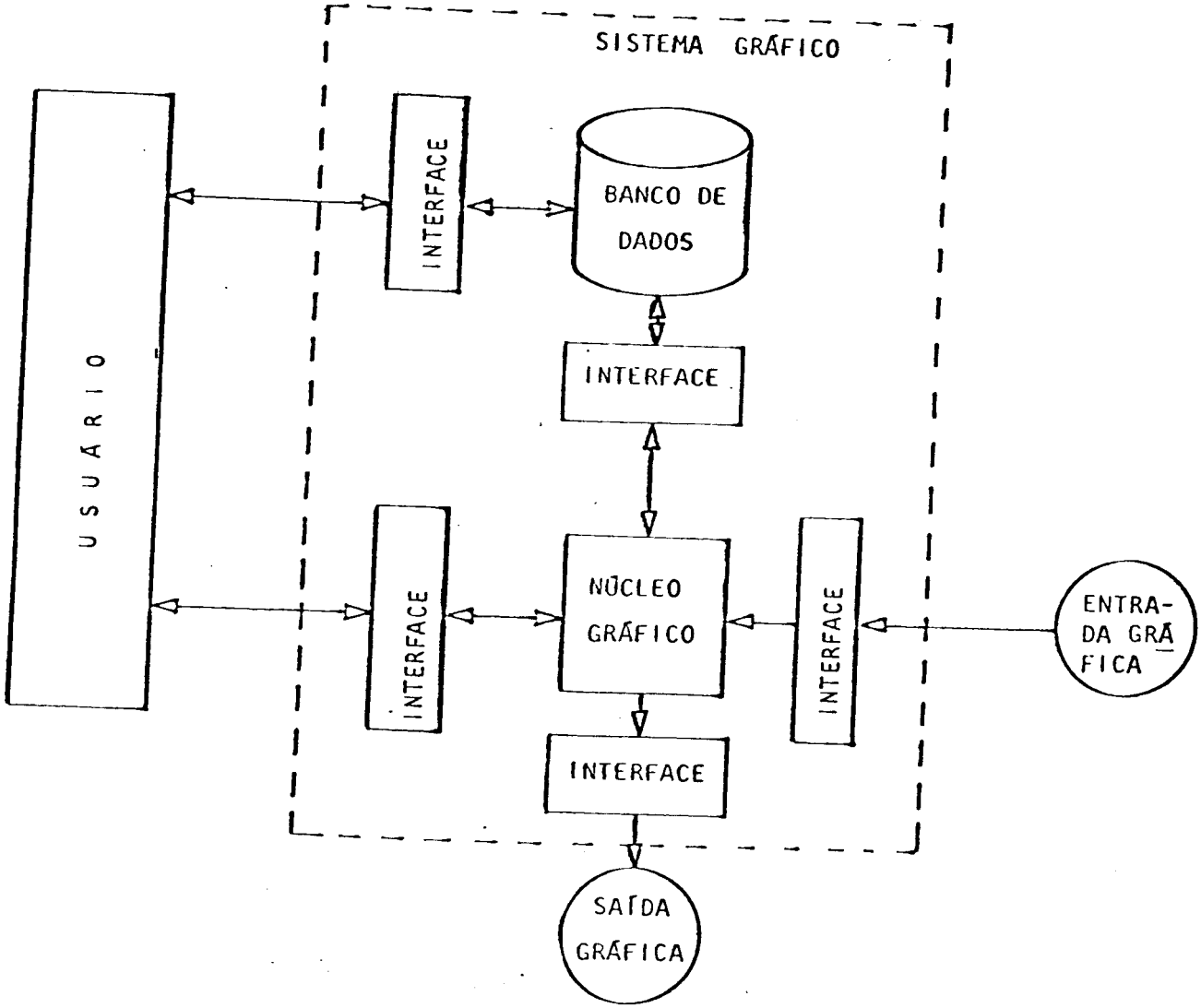



FIG. 4.1 - Modelo para a Implementação de um Sistema Gráfico

gráficas independentes da aplicação, que gerencia todos os recursos do sistema gráfico de base.

Neste capítulo desejamos modelar as necessidades da interface de saída. Isto a princípio é uma tarefa difícil, desde que ainda não foi definida uma estratégia para a implementação do núcleo. Contudo, acreditamos que as soluções apresentadas neste capítulo poderão ser tomadas como base para a definição de uma estratégia para a implementação do núcleo ou poderão mesmo ser adaptadas a um trabalho realizado com um completo desconhecimento deste. Na próxima seção, serão discutidos aspectos que caracterizem quais são os serviços que uma interface deve atender, destacando-se a independência versus a dependência do sistema gráfico de base em relação a dispositivos. Na seção 4.3, é apresentada a factibilidade da melhora do tempo de resposta do sistema pela utilização de um computador satélite. Na seção 4.4, são definidas as informações na fronteira núcleo/interface, especificando instruções de geração, funções de modificação e a necessidade de meios de consulta a tabela de estado da "workstation". Na última seção, um modelo funcional da interface é apresentado.

## 4.2 INDEPENDÊNCIA INTERNA

A independência de dispositivos que o pacote gráfico deve ter a nível do programador, foi apresentada várias vêzes nos capítulos anteriores. Agora, estamos interessados em determinar as partes independentes versus dependentes, internas ao pacote gráfico. Particularmente, deseja-se definir que serviços serão realizados na interface de saída. Num primeiro passo, para distinguir-se quais serviços são de competência de interface e quais do núcleo, deve ser realizado uma cuidadosa separação dos componentes que são inerentemente dependentes de dispositivos, daqueles que fazem parte do "software" comum a todos os dispositivos. Num segundo passo, dever-se-á resolver o problema dos serviços cuja classificação não é clara, ou seja, que podem tanto pertencer ao "software" comum, como podem ser considerados características do dispositivo. Por exemplo, a possibilidade de realizar-se transformações e "clipping" em "hardware", ou localmente, como veremos na próxima seção, levar-nos-á a ter situações confusas que nos obrigam a adotar algumas definições. A transformação geométrica, como é conceitualmente realizada no espaço de coordenadas normalizadas, poderá ser realizada ou no "software" comum do núcleo ou na interface. A transformação da "workstation" será considerada parte integrante da interface, seja ela executada por "software" ou "hardware". Aqui

vale a pena lembrar ao projetista da interface que,no caso de processadores que não realizem todo o conjunto de transformações e "clipping" necessárias por "hardware", praticamente nada se ganha ao realizar-se parte do processo de transformação e "clipping" por "hardware" e parte por "software": todo o processo executar-se-á na velocidade apresentada pelo "software".

A maior parte dos serviços realizados na interface, além das operações de transformação e "clipping", dar-se-ão pelo atendimento de características funcionais e formatação de dados do dispositivo. De modo geral, a interface ver-se-á frequentemente ocupada com manipulação de rotinas de geração / modificação de figuras para a superfície de visualização do dispositivo. O problema da definição da dependência de dispositivos pode ser minimizado se as informações da fronteira núcleo/interface forem bem definidas, e a interface do "software"comum para a interface de saída for bem projetada e documentada.

A interface do "software" comum para as rotinas da interface de saída pode tomar a forma de uma estrutura intermediária ou um conjunto de funções chamada pelo "software" comum. Alguns autores preconizam adoção de uma hierarquia no sistema, onde toda a troca de informação entre o núcleo do sistema e suas interfaces far-se-á através de uma codificação intermediária, que pode ser considerada como um conjunto de instruções de uma máquina abstrata / 1 /. Outros questionam a validade desta hierarquização, pois em um meio interativo isto adiciona um passo extra de armazenamento de informação, tendo um impacto negativo no tempo de resposta e na quantidade de memória adicional necessária / 9 /.

Aqui adotaremos uma solução intermediária. As rotinas da interface para a geração de figuras,tomarão a forma de uma estrutura de dados intermediária. Nesta estrutura de dados intermediária, toda a figura é armazenada num formato independente de dispositivos; a estrutura de dados é então traduzida para o formato requerido pelo dispositivo, pelasrotinas de geração em cada interface. Estas instruções estarão disponíveis numa lista linear de instruções, terminadas por um comando delimitador especial. Observe que esta codificação intermediária é bastante útil, desde que o armazenador de segmento pode armazená-la no mesmo formato. Assim, quando ocorrer uma função no programa de aplicação que descreva uma figura, ela é traduzida para a codificação intermediária e dirigida a todas as "workstation" ativas. Um armazenador de segmento é tratada como uma "workstation" e quando receber este código, podera armazená-lo neste mesmo formato.

Apesar da utilização de um armazenador de segmento, a possibilidade

de têrmos na interface um arquivo gráfico é uma ferramenta bastante útil. Dentro das rotinas de geração da interface de saída, devem existir rotinas de geração de código dependente para o terminal, que compilem instruções do mesmo para serem adicionadas ao arquivo gráfico. Deve existir uma rotina deste tipo para cada uma das instruções da estrutura de dados intermediária que representem primitivas de saída. Estas primitivas são passadas as rotinas de geração em coordenadas normalizadas, podendo passar por uma transformação geométrica na interface, e depois por uma transformação da "workstation". A partir daí, o gerador de código dependente para o terminal necessita rearranjar as coordenadas para ficar de acordo com o formato de instrução do terminal, e armazenar as instruções resultantes no arquivo gráfico.

A maneira como essas rotinas atuam, para o desenho de um segmento de reta é mostrado simplificadamente na figura 4.2. As transformações geométricas ocorrem conceitualmente no espaço de coordenadas normalizadas, e podem ou não ser agregadas a interface. Existem dois casos básicos onde a transformação é agregada à interface: ou o dispositivo possui todas as operações de transformação e "clipping" em "hardware", ou existe uma descentralização do dispositivo gráfico, discutida na próxima seção.

Se os pontos de entrada, representassem um círculo ao invés de um segmento de reta, caso o terminal não tenha esta primitiva em "hardware" é possível simulá-la por segmentos de reta. Isto pode ser realizado pela introdução na figura 4.2, de um bloco de simulação e uma realimentação para o processo "clipping", que será executada enquanto existirem segmentos de reta gerados pelo simulador. Isto é mostrado na figura 4.3. As rotinas de transformações geométricas, "window /viewport", "clipping" e de simulação de círculo quando realizadas por "software", são normalmente denominadas rotinas independentes, pois podem ser comuns a várias interfaces.

Desde que o nosso objetivo é tornar a interface de saída uma entidade autônoma no que diz respeito a ações de geração de primitivas, bem como a modificações para um terminal gráfico específico, a possibilidade da ocorrência de adiamento de primitivas gráficas na interface pode ser resolvido por algum mecanismo de armazenamento temporário na mesma. Para esse armazenamento, uma estrutura de armazenamento de dados gráficos, como é o arquivo gráfico, apresenta-se como uma boa solução. Sem a existência de um arquivo gráfico na interface não é possível determinar , por exemplo, se primitivas estão dentro ou fora de seg

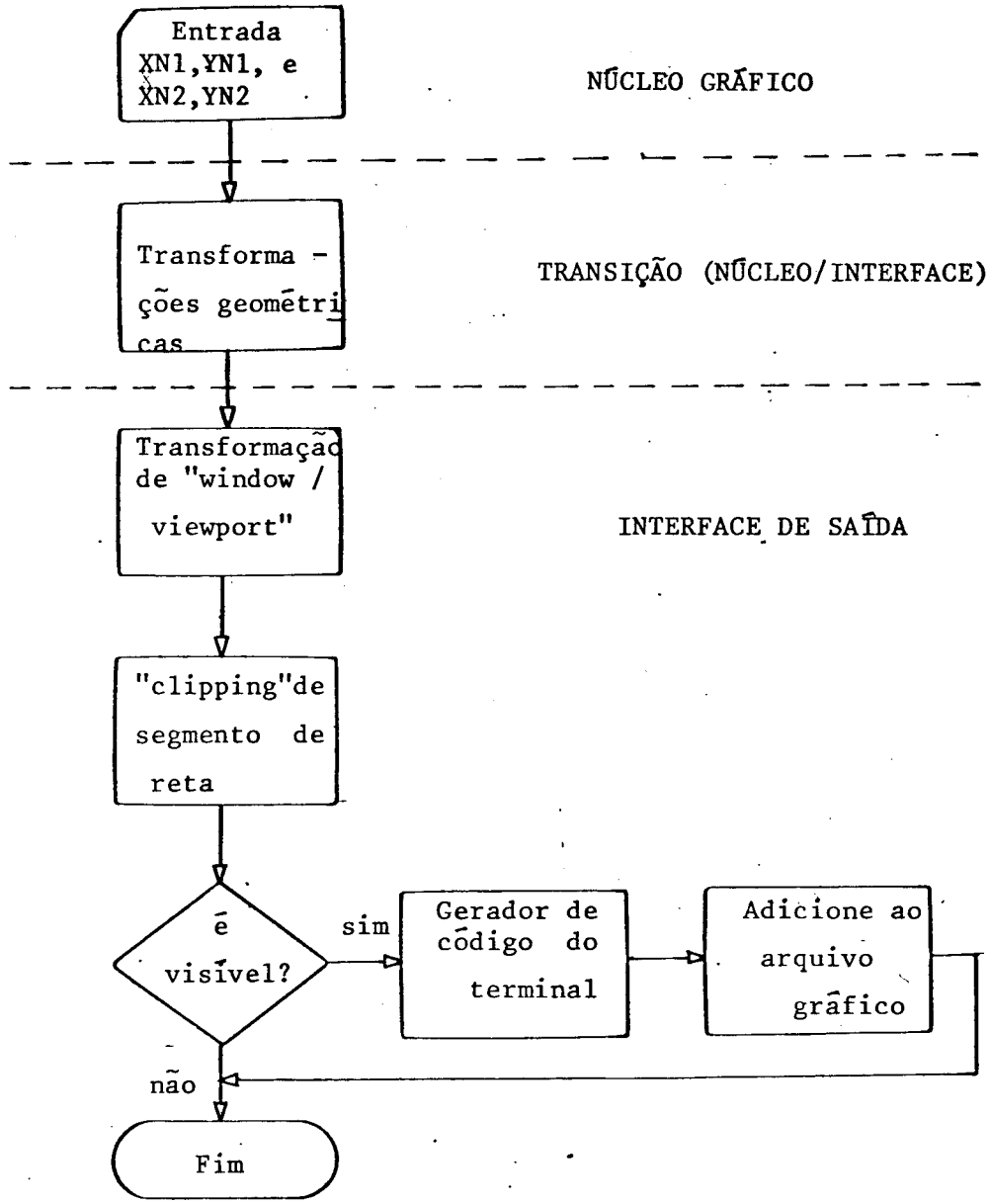



FIG: 4.2 - Fluxo de Operações para a Geração de um Segmento de Reta

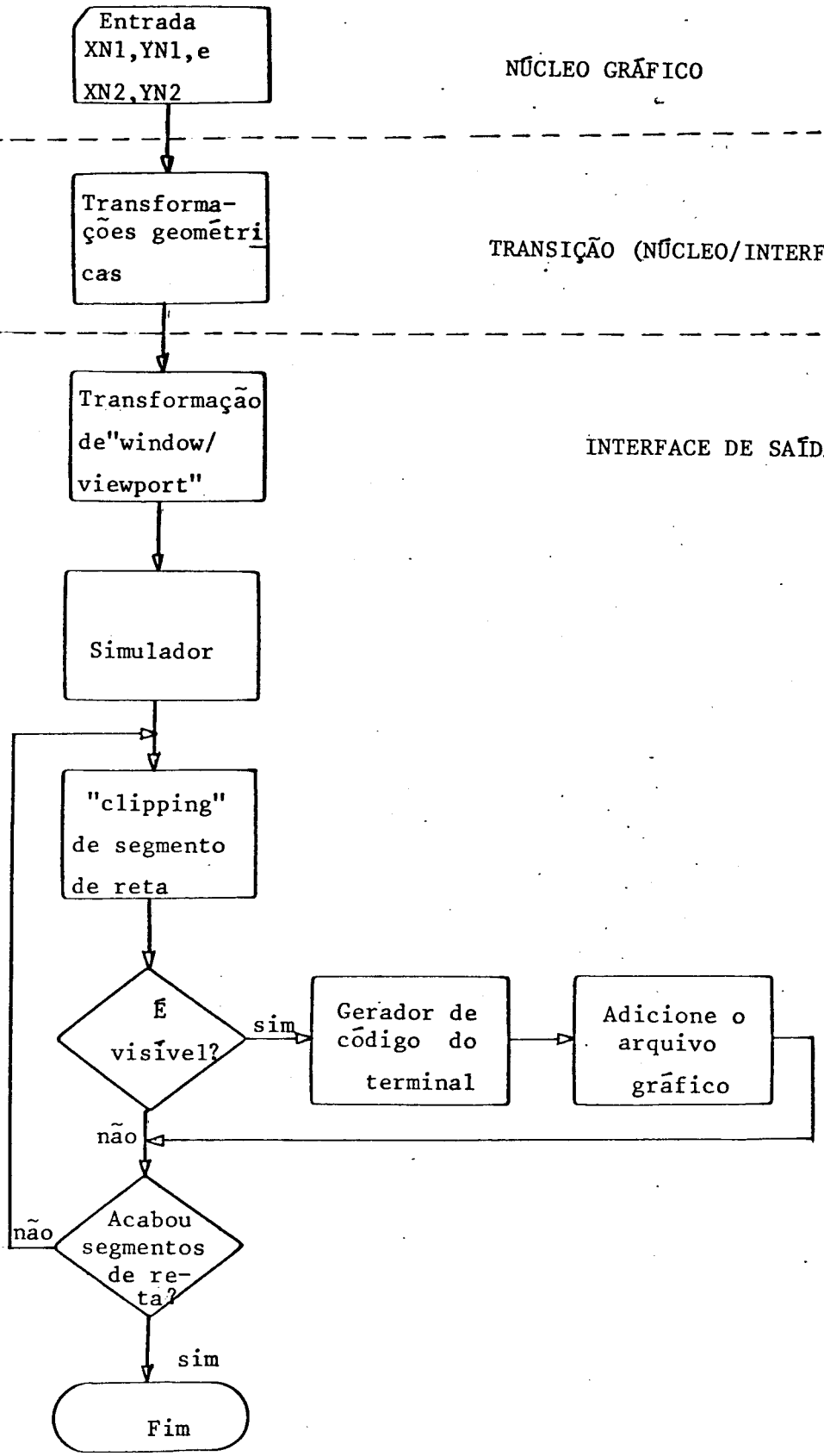



FIG. 4.3 - Fluxo de Operações para a Geração de um Círculo Simulado por Segmentos de Reta.

mentos, impossibilitando a ocorrência de regeneração implícita. No caso de modificações, que a princípio devem estar disponíveis na forma de um conjunto de funções, para permitir a ocorrência de modificações dinâmicas, a necessidade de uma estrutura como o arquivo gráfico é mais clara ainda.

## 4.3 DESCENTRALIZAÇÃO

O modo mais simples de organizarmos uma interface é deixarmos a maior parte desta implementada no hospedeiro, enviando-se ao processador gráfico apenas o programa gráfico da figura descrita. Toda vez que uma mudança na figura tiver de ser feita, uma cópia do programa gráfico atualizado deve ser mandado para o terminal, de forma a substituir a cópia corrente. Isto viola um dos requisitos básicos de sistemas gráficos interativos: o tempo de resposta torna-se intolerávelmente grande. Para resolver o problema descrito, incorpora-se ao processador gráfico um micro ou minicomputador, numa configuração conhecida como terminal gráfico satélite. Com a descentralização dos serviços realizados no hospedeiro, pode-se otimizar o desempenho do sistema.

Um sistema gráfico satélite tipico, é uma rede de três processadores, como mostrado na figura 4.4. O hospedeiro normalmente é um sistema computacional de multiprogramação ou de tempo compartilhado, e o satélite é usualmente dedicado para servir a um ou mais processadores gráficos. O "link" deve ser um meio de comunicação de alta velocidade.

Ao projetarmos uma interface, quando dispomos de um computador satélite, precisa-se decidir como serão divididas as tarefas de processamento entre os processadores, e como serão armazenados os dados nos dispositivos de memória. As possibilidades de distribuição de tarefas de processamento entre hospedeiro e o satélite são muitas, mas estamos interessados apenas no tipo de relacionamento denominado na literatura como função-fixa / 10 /. O satélite de função- fixa, é visto pela porção da interface que está implementada no hospedeiro como uma caixa preta cujo comportamento é essencialmente fixo e invariante. A parte da interface no hospedeiro controla o comportamento do satélite através de um conjunto de funções pré determinadas.

O projetista de interface deve fixar a divisão de trabalho, programando algumas funções no satélite, outras no hospedeiro. O programa de aplicação ao ser executado no hospedeiro produz grandes quantidades de informação que geram/mo

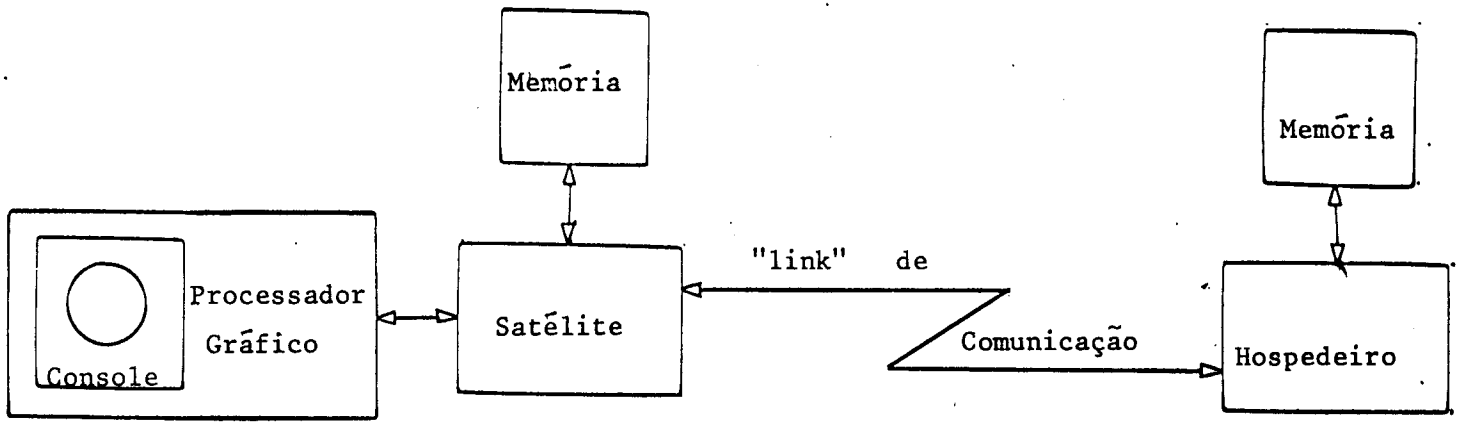



FIG: 4.4 - Configuração Típica de um Terminal Gráfico Satélite

dificam figuras que serão enviadas através do "link" de comunicação; tudo o que puder ser feito para minimizar o tamanho e a frequência de tais mensagens, pode melhorar o tempo de resposta do sistema.

Todo satélite precisa ter residente um mínimo de programas e dados. O dado básico necessário é o programa gráfico e os programas possíveis são: carregadores, modificadores, operadores do "link" de comunicação, e programas que tratam da interação com o usuário / 8 /. Pela adição de funções a este sistema mínimo, na forma descrita a seguir, o tamanho e a frequência de mensagens entre o hospedeiro e o satélite pode ser minimizado.

A compressão de dados durante a geração pode diminuir o tamanho e a frequência da informação gráfica enviada ao satélite, através de uma codificação eficiente. Por exemplo, para a realização da primitiva círculo, podem ser mandados junto a um comando identificador dessa primitiva, o ponto central e um ponto de sua circunferência, e então, se necessário, esta deve ser expandida no satélite numa série de segmentos de reta. Com respeito a modificações, a noção de um arquivo gráfico onde se possam distinguir segmentos pode ser utilizada para diminuir a quantidade de informação transmitida para o satélite. Por exemplo, se um computador satélite esta conectado a um vídeo armazenador, segmentos podem ser mantidos na memória do satélite. Quando um segmento precisa ser apagado da tela, este pode ser removido do arquivo gráfico localmente, evitando-se a retransmissão pelo hospedeiro de toda a figura restante.

Resumindo, existem muitas formas para minimizar a quantidade de dados transmitidos do hospedeiro para o satélite. No caso da compressão de dados, as partes da interface implementadas no hospedeiro e no satélite devem ser convenientemente estruturadas para obter-se o benefício pretendido. Observar-se-á na próxima seção que existe a necessidade de duplicar-se alguns dados da tabela de estado da "workstation" no computador satélite, se for desejada uma descentralização total. A possibilidade de realizar-se transformações geométricas localmente, deve ser uma opção a ser considerada.

## 4.4 INFORMAÇÕES DA FRONTEIRA NÚCLEO/INTERFACE

A determinação de um conjunto de informações que definam o comportamento da fronteira núcleo/interface de saída, esta baseada na análise da seção 4.2. Já nesta seção, haviamos nos referido a divisão da interface em roti

tinas de geração e modificação. A partir daí, foi apresentado a opção de têrmos a interface do "software" comum, para as rotinas de geração, por meio de instruções na forma de um código independente de dispositivos. Para as rotinas de modificação a interface é apresentada por meio de funções. Com esses dados foi realizado um estudo das funções do programa de aplicação e das tabelas do GKS, para determinação das instruções de geração e funções de modificação. Os dois grupos resultantes deste estudo são descritos a seguir.

O grupo de geração é composto por um conjunto de instruções que descrevem um objeto, seus atributos, e se estes estão fora ou dentro de segmentos. Estas instruções tomaram a forma de uma codificação intermediária independente de dispositivos. Esta codificação,cuja sintaxe é simples e livre de convenções rigidas de formato, servira para qualquer tipo de dispositivo gráfico - desde um "plotter" até um dispositivo vetorial com transformação em "hardware". A tabela 4.1., mostra este conjunto de instruções. Observe que a notação mnemônica para os comandos é introduzida somente para uma melhor compreensão das instruções. Como estas instruções são uma forma de comunicação interna do sistema, a cada mnemônico apresentado, corresponderá um código interno de operação. Estas instruções estarão disponíveis para todas as interfaces, através de uma lista linear de instruções, e a partir da interpretação deste arquivo e consulta a tabela de estado da "workstation", serão geradas instruções específicas de cada terminal. Nota-se na tabela 4.1., que o conjunto de instruções foi dividido em executáveis e não executáveis. As instruções não executáveis especificam se uma instrução executável esta fora ou dentro de segmentos e quais são seus atributos. As instruções executáveis são aquelas que descrevem objetos.

O comando ABRE SEGMENTO especifica que as instruções executáveis seguintes, estão dentro de um segmento. E permite que seja especificado um nome que identifique um segmento, usando até 8 caracteres. Os quatro primeiros no parâmetro NOME1,e os quatro últimos no parâmetro NOME2.

O comando NÚMERO PEN especifica que as instruções executáveis seguintes,que sejam afetadas pelo mesmo, estão associadas a um índice, que está definido no parâmetro VALOR.

O comando NÚMERO TEXT especifica que as instruções executáveis seguintes, que sejam afetadas pelo mesmo, estão associadas a um índice, que está definido na parâmetro VALOR.

O comando TAMANHO E ESPAÇAMENTO DE TEXTO especifica que as

TABELA 4.1 - INSTRUÇÕES DE GERAÇÃO

| COMANDO | PARÂMETROS |
|---|---|
| / NÃO EXECUTÁVEIS/ | - |
| ABRE SEGMENTO | NOME1, NOME2 |
| NÚMERO PEN | VALOR |
| NÚMERO TEXT | VALOR |
| TAMANHO E ESPAÇAMENTO DE TEXTO | VTX,VTY,HLX,HLY,EX,EY |
| IDENTIFICADOR DE PICK | VALOR |
| TAMANHO DE MARCADOR | VTX,VTY,HLX,HLY |
| FECHA SEGMENTO | - |
| /EXECUTÁVEIS/ | - |
| POLILINHA | N,X1,Y1,..,XN,YN |
| POLIMARCADOR | N,X1,Y1,..,XN,YN,TIPO |
| GERAL | N,X1,Y1,..,XN,YN,TIPO |
| TEXTO | X,Y,N, SEQUÊNCIA DE CARACTERES |
| ÁREA | N,X1,Y1,..,XN,YN |
| MATRIZ DE PIXELS | XE,YE,XD,YD,M,N,MATRIZ DE CORES POR LINHA (MXN) |

instruções executáveis seguintes, que sejam afetadas pelo mesmo, estão associadas às características descritas por seus parâmetros. Os parâmetros VTX e VTY, especificam a direção da linha vertical e o tamanho do caracter; HLX e HLY especificam a direção da linha horizontal e a largura do caracter; EX e EY, especificam o espaçamento entre caracteres. Os valores de todos os parâmetros são relativos ao ponto inicial do caracter.

O comando IDENTIFICADOR DE PICK especifica que as instruções executáveis seguintes, que sejam afetadas pelo mesmo, estão associadas a um identificador, que está definido no parâmetro VALOR.

O comando TAMANHO DE MARCADOR especifica que as instruções executáveis seguintes, que sejam afetadas pelo mesmo, estão associadas as características descritas por seus parâmetros. Os parâmetros VTX e VTY, especificam a direção da linha vertical e o tamanho do marcador; HLX e HLY, especificam a direção da linha horizontal e a largura do marcador. Os valores de todos os parâmetros são relativos ao canto inferior esquerdo do marcador.

O comando FECHA SEGMENTO especifica que o último segmento aberto foi fechado.

O comando POLILINHA especifica um conjunto de linhas conectadas. O parâmetro N especifica o número de pontos disponíveis.

O comando POLIMARCADOR especifica um conjunto de marcadores do mesmo tipo. O parâmetro N especifica o número de pontos disponíveis e o parâmetro TIPO especifica o tipo do símbolo marcador, este deve aparecer centrado nos pontos dados.

O comando GERAL especifica um conjunto de pontos que descrevem alguma primitiva não convencional. O parâmetro N especifica o número de pontos disponíveis. O parâmetro TIPO especifica o tipo da primitiva.

O comando TEXTO especifica uma sequência de caracteres. Os parâmetros X e Y, indicam o ponto inicial do primeiro caracter e o parâmetro N o número de caracteres disponíveis.

O comando ÁREA especifica um polígono que é fornecido. O parâmetro N especifica o número de pontos disponíveis.

O comado MATRIZ DE PIXELS especifica uma matriz retangular de células. Os parâmetros XE e YE, definem o ponto inferior esquerdo da matriz; XD e YD, definem o ponto superior direito; M define o número de linhas; N define o número de colunas; e uma matriz de índice "pen" armazenada por linha (MxN) indica

as características de cada célula.

Sempre que necessário, os parâmetros dos comandos da tabela 4.2 devem ser fornecidas a interface em coordenadas normalizadas.

O grupo de modificação é composto por funções que possam realizar modificações no estado da interface. As funções de modificação estão mais intimamente ligadas à implementação do núcleo gráfico do que as instruções de geração. Por isso, sempre que acharmos necessário, ao se explicar o significado do comando, enfatizar-se-á o que a interface realiza e/ou as características peculiares que a mesma deve possuir. A tabela 4.2, mostra os comandos e parâmetros das funções de modificação. Em princípio, estes comandos estarão disponíveis através de subrotinas. Outra observação de relevância a se fazer, é a adição de informações à tabela de descrição da "workstation" especificando se a interface tem capacidade de armazenamento e transformação local de segmento.

O comando INICIALIZAÇÃO especifica que a interface deve ter todos os seus estados inicializados.

O comando LIMPEZA DA TELA especifica que a superficie de visualização do dispositivo gráfico deve ser inicializada.

O comando ATUALIZAÇÃO especifica que todos os processos apresentados a interface que foram adiados devem ser realizados; é necessário que a interface tenha capacidade de armazenamento.

O comando REMOÇÃO especifica que todas as informações do segmento, especificado pelos parâmetros NOME1 e NOME2, devem ser apagadas.

O comando MUDANÇA DE NOME especifica que todas as informações associadas ao nome de segmento especificado nos parâmetros VEME1 e VEME2, devem ser associadas ao seu novo nome, especificado nos parâmetros NOME1 e NOME2.

O comando TRANSFORMAÇÃO DE SEGMENTO especifica, para interfaces que permitam modificações dinâmicas ou locais, o nome do segmento e a matriz de transformação armazenada por linha (2x3) que deve ser aplicada ao segmento.

O comando MUDANÇA DE VISIBILIDADE especifica que foi mudado o atributo de visibilidade do segmento de nome explicitado nos parâmetros NOME1 e NOME2. O parâmetro B especifica se o segmento é visível (B=1) ou não (B=0).

TABELA 4.2 - FUNÇÕES DE MODIFICAÇÕES

| COMANDO | PARÂMETROS |
|---|---|
| INICIALIZAÇÃO | - |
| LIMPEZA DA TELA | - |
| ATUALIZAÇÃO | - |
| REMOÇÃO | NOME1,NOME2 |
| MUDANÇA DE NOME | VEME1,VEME2,NOME1,NOME2 |
| TRANSFORMAÇÃO DE SEGMENTO | NOME1,NOME2,MATRIZ DE TRANSFORMAÇÃO |
| MUDANÇA DE VISIBILIDADE | NOME1,NOME2,B |
| MUDANÇA DE DETETABILIDADE | NOME1,NOME2,B |
| MUDANÇA DE PISCAMENTO | NOME1,NOME2,B |
| MUDANÇA DE PRIORIDADE | NOME1,NOME2,R |
| MUDANÇA DE PEN | VALOR |
| MUDANÇA DE TEXT | VALOR |
| TRANSFORMAÇÃO DA WORKSTATION | - |

O comando MUDANÇA DE DETETABILIDADE especifica que foi mudado o atributo de visibilidade do segmento de nome explicitado pelos parâmetros NOME1 e NOME2. O PARÂMETRO B especifica se o segmento é detetável (B=1) ou não (B=0).

O comando MUDANÇA DE PISCAMENTO especifica que foi mudada o atributo de piscamento do segmento de nome explicitado pelos parâmetros NOME1 e NOME2. O PARÂMETRO B especifica se o segmento vai piscar (B=1) ou não (B=0).

O comando MUDANÇA DE PRIORIDADE especifica que foi mudada a prioridade do segmento de nome especificado pelos parâmetros NOME1 e NOME2.O parâmetro R especifica sua nova prioridade.

O comando MUDANÇA DE PEN especifica que foram modificadas as características do atributo número "pen" explicitado pelo parâmetro VALOR.

O comando MUDANÇA DE TEXT especifica que foram modificadas as características do atributo número "text" explicitado pelo parâmetro VALOR.

O comando TRANSFORMAÇÃO DA WORKSTATION especifica, para interfaces que permitam modificações dinâmicas, que a transformação corrente foi mudada.

Tanto o grupo de geração, quanto o de modificação, necessitam prover a interface de dados adicionais. Estes dados, em ambos os casos, devem ser obtidos da tabela de estado da"workstation". Particularmente, os atributos de "pen", "text" e da"workstation", além do estado de adiamento devem ser obtidos nesta tabela. Caso a parte da interface que as necessite esteja implementada no hospedeiro, essas informações podem ser obtidas através das funções de inquisição do estado do GKS, que podem ser acessadas pelo programa de aplicação.

A possibilidade de se enviar mensagens também deve ser considerada. Esta característica da interface não foi discutida por não ter nenhum vínculo com os processos de geração/modificação. Esta poderia ser implementada através de uma subrotina cujo parâmetro permita enviar uma mensagem de quatro caracteres para a interface. Esta caracterísitica pode ser facilmente anexada a interface caso o dispositivo gráfico de saída funcione também como terminal alfanumérico.

## 4.5 MODELO FUNCIONAL DA INTERFACE

Importunamente, não é uma tarefa fácil descrever-se um modelo funcional geral da interface de saída, desde que existe um conjunto muito grande de dispositivos com características diversas. Mesmo que restrinjamos o modelo a um subconjunto dos dispositivos existentes, este deve ser funcionalmente modular para atender a dispositivos distintos, como é o "plotter" comparado a um vídeo vetorial regenerativo. Considerando estes problemas, um modelo funcional é proposto e é mostrado na figura 4.5 / 11 /.

A restrição básica do modelo proposto, é a de basear-se na existência de um arquivo gráfico com a característica peculiar de ser capaz de permitir que qualquer de seus elementos, grupos de instruções do dispositivo, possam: ser divididos em partes lógicas denominadas segmento, ser mudados, ser removidos, ou ter uma característica adicional introduzida. O arquivo gráfico pode existir ou não dependendo das características do dispositivo utilizado pelo sistema. Por exemplo, para um vídeo regenerativo a presença de um arquivo gráfico é imprescindível e existirá um arquivo gráfico real. Para um vídeo armazenador sua presença é facultativa, mas conveniente e, no modelo existirá um arquivo gráfico virtual. Para um "plotter", o arquivo gráfico poderá não existir.

O bloco interpretador é responsável pela interpretação dos códigos independentes de dispositivos descritos na tabela 3.1. Para a realização desta tarefa, deverão existir meios de consulta a tabela de estado da "workstation". Este bloco deve ainda ter um conhecimento profundo da organização dos dados no arquivo gráfico, para que possa ser montada uma estrutura de dados denominada mapa de correlação.

No mapa de correlação estarão contidas estruturas de dados que permitam a realização de modificações dinâmicas e/ou processos interativos/ 8 /.

O bloco modificador é responsável pelo atendimento das funções descritas na tabela 3.2. Para a realização desta tabela, este bloco deverá ter acesso facilitado ao mapa de correlação para atualizar/obter dados. Deverá ainda ter meios de consulta a tabela de estado da "workstation", além de acesso ao interpretador e ao arquivo gráfico.

O interpretador deverá aceitar todas as instruções descritas na tabela 3.1, e rejeitar aquelas que a interface não tem capacidade para processar.

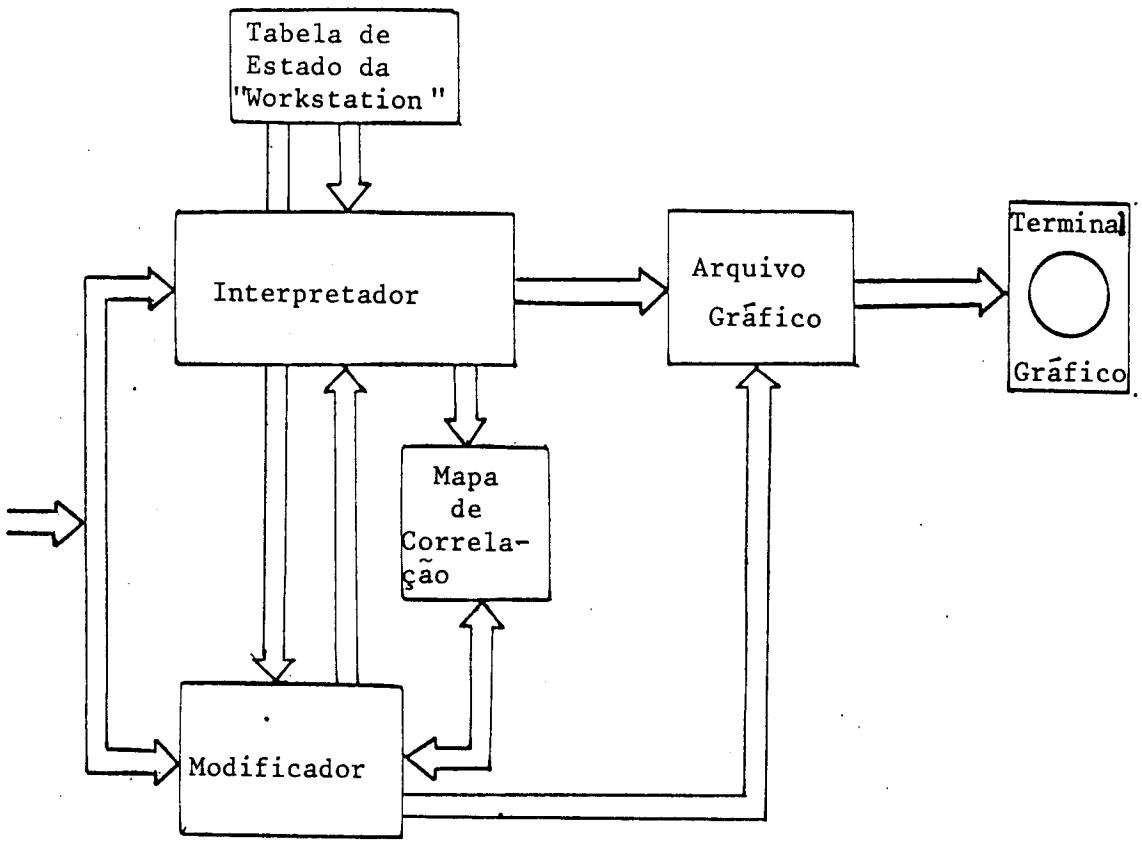



FIG: 4.5 - Modelo Funcional da Interface de Saída

Esta rejeição não será necessária se na tabela de descrição da "workstation" for acrescentada informação descreyendo quais instruções estão implementadas na interface; cada interface recebe apenas instruções que ela tem capacidade de processar. Para funções de modificação da interface, caso seja necessário, deve ser adotada esta mesma política.

# CAPÍTULO 5

# INTERFACE PARA UM VÍDEO VETORIAL REGENERATIVO

Neste capítulo é apresentado, como exemplo de aplicação do modelo funcional da interface de saída, a implementação de uma interface para um vídeo vetorial regenerativo. Na primeira seção, é apresentado um panorama dos recursos disponíveis. Na seção 5.2, são apresentadas as principais características desta interface. Na seção 5.3, são apresentados os algoritmos básicos das rotinas independentes da interface: "window/viewport", "clipping", círculo e arco de círculo. Na seção 5.4, é apresentado a estrutura do arquivo gráfico. Na seção 5.5, são apresentados como e com quais características foram implementados os atributos das instruções executáveis e de segmento. Na seção 5.6, é apresentado a estrutura de dados do mapa de correlação e suas peculiaridades. Na seção 5.7, são apresentados as ações tomadas pelas rotinas dos comandos de geração implementados. Na última seção, são apresentados as ações básicas tomadas pelas rotinas dos comandos de modificação implementados.

## 5.1 RECURSOS DISPONÍVEIS

Para o exemplo de implementação de uma interface de saída para um sistema gráfico de base utilizaremos um terminal gráfico satélite GT- 4Ø / 12,13 /. O GT-4Ø é um terminal gráfico do tipo vetorial regenerativo, acoplado a um minicomputador PDP-11/Ø5. Este terminal constitui-se basicamente das seguintes partes: unidade central de processamento, processador gráfico, "light pen", interface de comunicação, memória, e monitor em ROM. A unidade central de processamento é um processador com palavras de 16 bits, com um conjunto padrão de instruções do PDP-11 e 8K palavras de memória. O GT-4Ø pode ser visto como um computador que possui dois processadores programáveis separadamente. A unidade central de processamento colhe instruções da memória e as executa lendo ou modificando os dados armazenados na memória. O processador gráfico colhe instruções e dados em posições contíguas da memória que forma o programa gráfico; a comunicação processador gráfico - memória é unidirecional.

A ligação entre o GT-4Ø e o hospedeiro é feita por uma interface assíncrona que transfere e recebe 8 bits via "modem". A velocidade de transmis

são é de 1200 b.p.s. O hospedeiro (PDP-10) é um sistema computacional multiprogramado que opera em tempo compartilhado.

Por ser um computador dedicado a uma aplicação específica, o PDP-11/05 não possui facilidades como sistema operacional e compiladores. Porém, pode ser programado em "assembly" pois o hospedeiro possui o montador necessário. O GT-40 tem um monitor em "hardware"que inicializa seu funionamento como terminal. Após esta inicialização, o hospedeiro pode transferir o programa montado em linguagem de máquina para a memória do terminal por meio de um carregador. Com estes recursos, o hospedeiro dispõe de um pacote gráfico para a utilização do terminal gráfico. Este pacote consiste de dois programas, um em"assembly", a ser carregado no terminal, outro em FORTRAN, uma série de subrotinas que o usuário utiliza no seu programa para compor a informação gráfica. Desde logo, observou-se que para resolvermos o nosso problema era necessário projetar uma interface sem a utilização dos recursos oferecidos pelo pacote existente. Dentre os motivos que nos levaram a esta conclusão, incluem-se: impossibilidade de se atender entradas no modo "event", lentidão no atendimento a outros processos interativos, má utilização de memória, e a complexidade e, até mesmo a impossibilidade, de realizar- se modificações dinâmicas.

O processador gráfico (GT-40) apresenta um conjunto variado de instruções. Porém estamos interessados em descrever apenas um subconjunto dessas instruções e suas características principais, essenciais para compreensão do método de expansão utilizado pelo interpretador, bem como para o entendimento de como ocorrem algumas modificações. Dividiu-se este subconjunto de instruções em três grupos básicos, onde são apresentados os mnemônicos das instruções e as características relevantes.

Grupo 1: PONTO
VETOR LONGO
VETOR CURTO
CARACTER

Condições Gerais:

- As instruções vetor longo, vetor curto e caracter são relativas.
- Após a definição de uma instrução relativa, pode seguir-se qualquer número de dados referentes a mesma
- A visibilidade é uma propriedade dos dados que seguem estas instruções, exceto para os dados da instrução caracter.

. Os atributos apresentados a seguir podem se agregados a qualquer uma das instruções desse grupo. Após o agregamento a uma instrução de qualquer uma das características apresentadas, todas instruções posteriores a obedecem, até que ocorra uma redefinição desta.

. Intensidade — Permitem-se oito níveis de instensidade

. Tipo de Linha — Permitem-se quatro tipos de linhas:
  . Sólido
  . Tracejada longo
  . Tracejada Curta
  . Ponto - traço

. Detetabilidade

. Piscamento

Grupo 2: PULO

Condições Gerais:

. O processamento gráfico desvia para o endereço especificado pelo dado seguinte a instrução.

Grupo 3: STATSA

Condições Gerais:

. O registro A do processador gráfico é carregado com informações contidas nesta instrução.

. Nesta instrução pode ser especificado o tipo de caracter como sendo padrão ou itálico.

## 5.2 CARACTERÍSTICAS GERAIS DA INTERFACE

### 5.2.1 Comandos implementados

Pela observação das características funcionais do GT-4Ø e das informações na fronteira núcleo/interface, determinaram-se os comandos que foram implementados. O resultado deste estudo está apresentado nas tabelas 5.1 e 5.2, as quais são cópias da coluna de comandos das tabelas 4.1 e 4.2, apresentadas na se

TABELA 5.1 - INSTRUÇÕES DE GERAÇÃO UTILIZADAS

| COMANDOS UTILIZADOS PELO VÍDEO VETORIAL | |
|---|---|
| /NÃO EXECUTÁVEIS/ | |
| ABRE SEGMENTO | X |
| NÚMERO PEN | X |
| NÚMERO TEXT | X |
| TAMANHO E ESPAÇAMENTO DE TEXTO | - |
| IDENTIFICADOR DE PICK | X |
| TAMANHO DE MARCADOR | - |
| FECHA SEGMENTO | X |
| /EXECUTÁVEIS/ | |
| POLILINHA | X |
| POLIMARCADOR | X |
| GERAL | X |
| TEXTO | X |
| ÁREA | X |
| MATRIZ DE PIXELS | X |

X = Utilizado

\- = Não utilizado

TABELA: 5.2 - FUNÇÕES DE MODIFICAÇÃO UTILIZADAS

| COMANDOS UTILIZADOS PELO VÍDEO VETORIAL | |
|---|---|
| INICIALIZAÇÃO | X |
| LIMPEZA DA TELA | - |
| ATUALIZAÇÃO | X |
| REMOÇÃO | X |
| MUDANÇA DE NOME | X |
| TRANSFORMAÇÃO DE SEGMENTO | - |
| MUDANÇA DE VISIBILIDADE | X |
| MUDANÇA DE DETETABILIDADE | X |
| MUDANÇA DE PISCAMENTO | X |
| MUDANÇA DE PRIORIDADE | - |
| MUDANÇA DE PEN | X |
| MUDANÇA DE TEXT | X |
| TRANSFORMAÇÃO DA WORKSTATION | - |

X = Utilizado

- = Não utilizado

ção 4.4, com a adição de mais uma coluna indicando quais comandos foram implementados.

### 5.2.2 Distribuição de tarefas

Desde que existem dois processadores, é preciso definir a distribuição de tarefas entre o hospedeiro e o computador satélite, e quais os critérios que foram utilizados para essa distribuição.

O hospedeiro, no que a tange ao interpretador, ficou basicamente responsável pela geração de um arquivo com a descrição da figura em instruções dependentes da "workstation", e pela criação de todos os dados do mapa de correlação. No que tange à modificação, o hospedeiro ficou basicamente responsável pela obtenção/atualização de dados do mapa de correlação.

O satélite, no que tange ao interpretador, ficou basicamente responsável pelo carregamento do programa gráfico, montado no hospedeiro, na estrutura do arquivo gráfico existente na memória do terminal. No que tange à modificação, ficou basicamente responsável pelas partes das rotinas modificadoras que atuam sobre o arquivo gráfico.

O principal critério utilizado para adotar-se esta distribuição foi a pequena quantidade de memória existente no satélite. As rotinas no hospedeiro forma implementadas em FORTRAN e as rotinas do satélite am "assembly" do PDP 11.

Os algoritmos que descrevemos nas seções seguintes, na maioria dos casos, estão simplificados com o objetivo de não apresentar-se detalhes dos algoritmos, mas sim, os princípios fundamentais que os governam.

### 5.2.3 Informações da tabela de estado da workstation

As informações da tabela de estado da "workstation" são necessárias para o perfeito funcionamento da interface, são obtidas através das funções de inquisição do GKS. O nome completo das funções de inquisição utilizadas e seus dados significativos, são apresentados a seguir:

.INQUIRE PREDEFINED PEN ATTRIBUTES - Dado um índice "pen", as características deste índice são devolvidas.

.INQUIRE PREDEFINED TEXT ATTRIBUTES - Dado um índice "text", as caracteristi

cas deste índice são devolvidas.

.INQUIRE WORKSTATION STATE - Dado o identificador da "workstation", o estado de adiamento é devolvido.

.INQUIRE WORKSTATION TRANSFORMATION - Dado o identificador da "workstation", os parâmetros para a transformação de coordenadas são devolvidos.

### 5.2.4 Modelo geral

Ao realizarmos qualquer projeto, necessita-se de um plano geral que descreva as principais características do trabalho a ser realizado, Na seção 4.5, apresentou-se um modelo funcional sobre o qual o projetista da interface deve basear-se. O modelo funcional configurado para as necessidades desta interface é mostrado na figura 5.1. Nesta observa-se que o fluxo de atendimento das instruções do interpretador tomam dois caminhos diferentes, um para atender as instruções executáveis, outro para as não executáveis. Três blocos de rotinas independentes são vistos nesta figura. O bloco de "window/viewport" realiza a transformação de coordenadas normalizadas para coordenadas do dispositivo. O bloco simulador visa atender as primitivas especiais da instrução GERAL; nesta implementação existem as primitivas especiais círculo e arco de círculo. O bloco de "clipping" realiza o corte das informações que ultrapassem o"viewport"especificado. Do lado do atendimento das instruções não executáveis, o bloco de controle atua sobre o gerador de código dependente da "workstation", informando-lhe quais os valores dos últimos atributos e se as instruções executáveis estão fora ou dentro de segmento.

Uma descrição pormenorizada do bloco modificador e do mapa de correlação é dada mais adiante, desde que para uma boa compreensão de suas características, necessita-se um bom conhecimento dos detalhes de implementação da estrutura do arquivo gráfico e dos atributos.

## 5.3 ROTINAS INDEPENDENTES

### 5.3.1 "Window/Viewport"

Utiliza-se uma "window" para descrever-se o que desejamos que seja visível numa figura e o "viewport" para especificarmos a escala e a posição

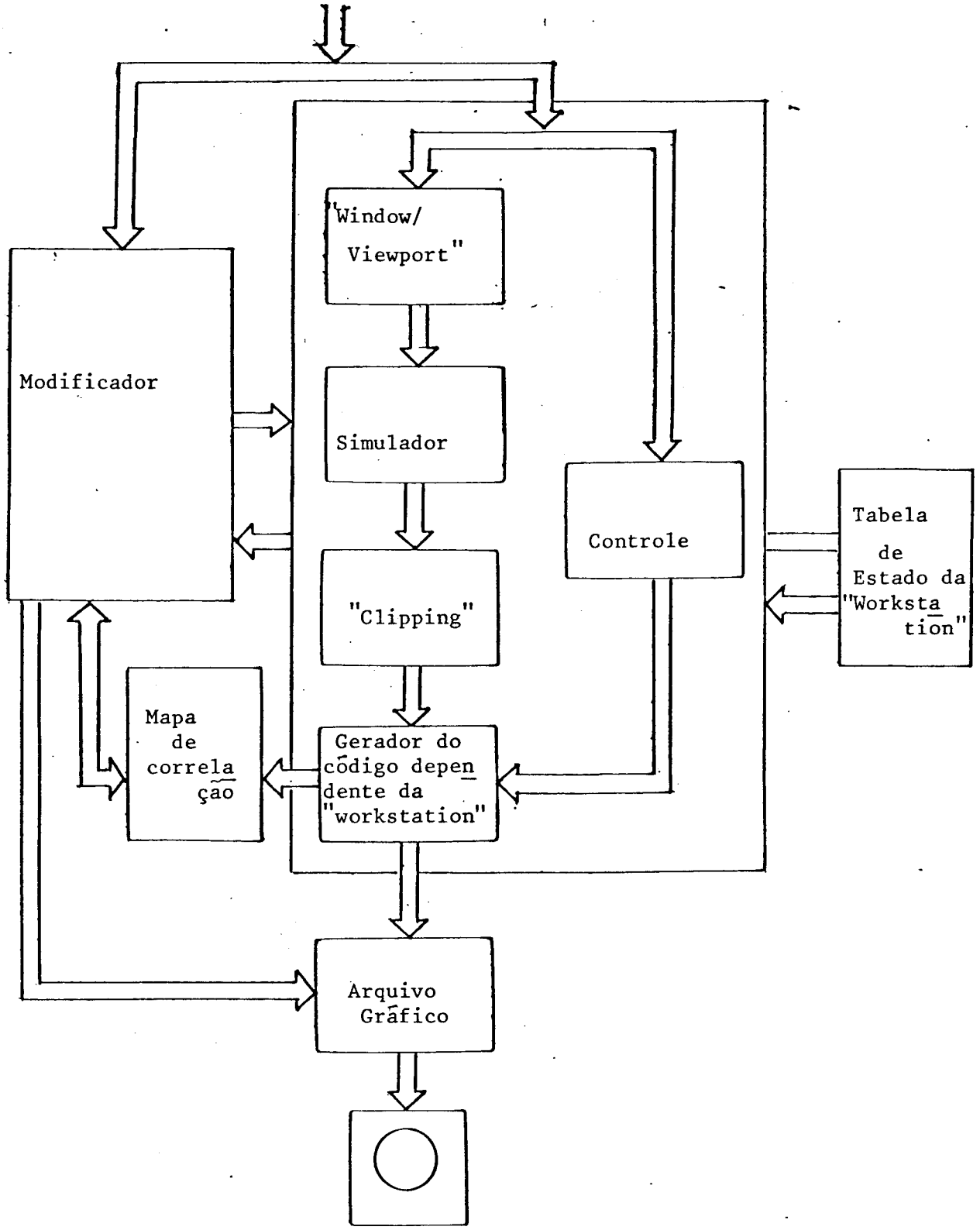



FIG: 5.1- Modelo Funcional da Interface para o Vídeo Vetorial Regenerativo GT-4Ø

em que a "window" será mostrada na superfície de visualização. Neste mapeamento pode ocorrer uma transformação de coordenadas. Apesar de um "window/viewport" permitir uma mudança de escala e posição, pela filosofia de "viewport" definida pelo GKS somente é permitida a mudança de escala na "workstation".

A transformação que é aplicada a cada ponto é realizada da maneira descrita a seguir. Sejam as bordas da "window" dadas por WXE,WXD,WYB e WYC como mostrado na figura 5.2, todos os pontos devem estar em algum sistema de coordenadas. Sejam as bordas do "viewport", para a qual será mapeada a "window", dadas por VXE,VXD,VYB e VYC, todas em algum sistema de coordenadas.

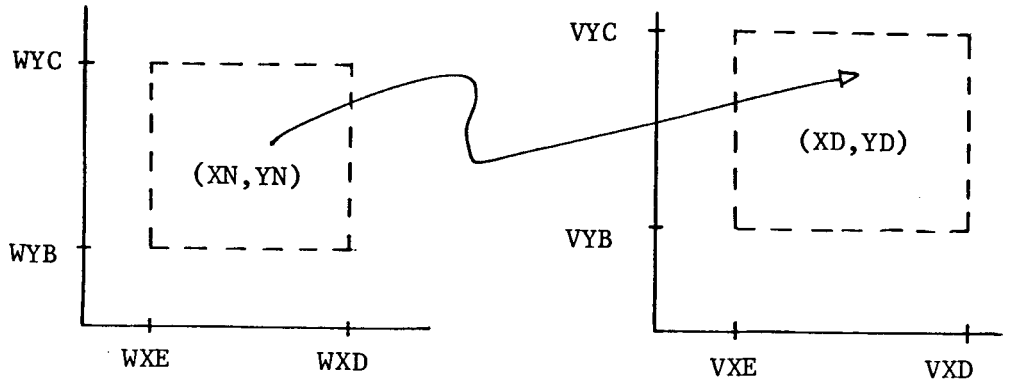



FIG: 5.2 - A Transformação de "Window/Viewport"

Se o ponto (XN,YN) deve ser transformado no ponto (XD,YD), as equações do mapeamento são:

$$XD = \frac{VXD - VXE}{WXD - WXE} * (XN - WXE) + VXE$$

$$YD = \frac{VYC - VYB}{WYC - WYB} * (YN - WYB) + VYB$$

### 5.3.2 "Clipping"

No processo de corte ("clipping") das partes da figura que estão fora da tela, definida como qualquer espaço retangular paralelo ao eixo de coordenadas, dois tipos de corte são de relevância:

- Corte de ponto
- Corte de reta

O corte de reta utiliza o algoritmo de corte de ponto. O algoritmo de corte de ponto é utilizado quando trabalhamos com caracteres. Usualmente, calcula-se a diagonal do caracter e caso os seus pontos extremos estejam dentro da tela o caracter é mostrado; caso contrário, é rejeitado. Existem vários algoritmos para corte de reta na literatura, mas abordaremos apenas o que foi implementado que, não só acha o ponto de interseção da reta com a tela rapidamente, como também rejeita a reta invisivel com a mesma rapidez / 9 /.

O algoritmo é subdividido em duas partes: na primeira, determina-se se a reta está totalmente contida na tela - caso não esteja ela pode ser trivialmente rejeitada; na segunda, a reta está parcialmente contida na tela e é dividida em duas partes - uma delas é trivialmente rejeitada e a outra é submetida novamente a testes.

| | | |
|---|---|---|
| 1001 | 1000 | 1010 |
| 0001 | TELA 0000 | 0010 |
| 0101 | 0100 | 0110 |

FIG: 5.3 - Nove regiões em que a Figura Não Cortada é Dividida.

O teste de rejeição é implementado estendendo-se as bordas da tela especificada, de forma a dividirmos o espaço ocupado pela figura não cortada em nove regiões. Isto é mostrado na figura 5.3.

Cada uma destas regiões tem associado um código de quatro bits e aos pontos extremos da reta são atribuídos códigos de acordo com a região em que ele se localiza. Estes códigos indicam se um ponto está dentro ou fora da tela. O significado dos quatros bits do código é especificado a seguir:

Primeiro Bit : O ponto está acima da borda superior da tela.
Segundo Bit : O ponto está abaixo da borda inferior da tela.
Terceiro Bit : O ponto está à direta da borda direita da tela.
Quarto Bit : O ponto está à esquerda da borda esquerda da tela.

É óbvio que se o quatro bits para dois pontos extremos é zero, a reta está totalmente contida na tela. Embora não seja óbvio, pode-se mostrar que se a interseção lógica de dois códigos é diferente de zero, a reta está totalmente fora da tela. Se a reta não puder ser definida pelos dois testes anteriores, então ela precisa ser subdividida. Um método simples de subdivisão, é o de acharmos o ponto da interseção da reta com uma das bordas da tela e retirarmos a parte que se encontra fora da tela. Por exemplo, a reta AB da figura 5.4, é inicialmente submetidas aos testes, e logo é observado que ela precisa ser subdividida em C, e a parte AC é trivialmente rejeitada. A nova reta CB é resubmetida aos testes, mais uma vez precisamos subdividi-la em D, e a reta CD é trivialmente rejeitada. A reta DB encontra-se totalmente contida na tela, e será devolvida pelo algoritmo.

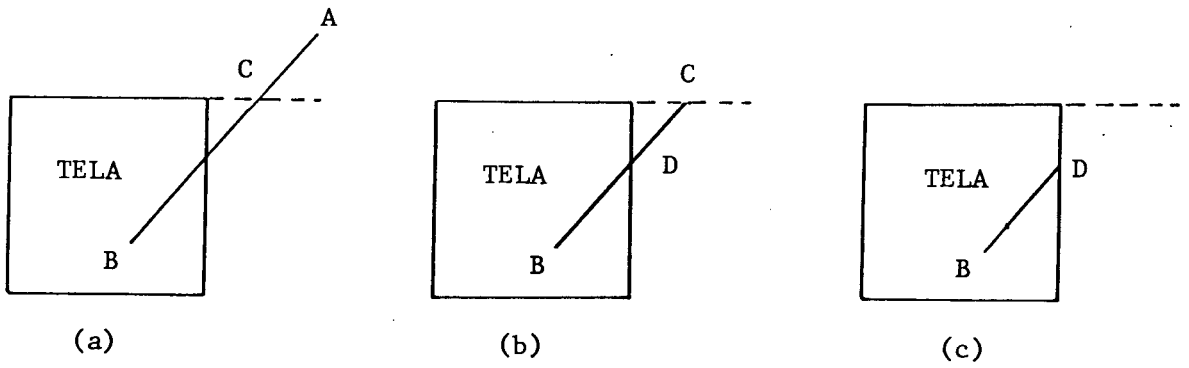



FIG: 5.4 - Exemplo de Aplicação do Algoritmo de Corte de Reta.

### 5.3.3 Círculo e arco de círculo

Para a representação matemática de uma curva, pode-se utilizar uma equação paramétrica ou não paramétrica. Na forma paramétrica cada coordenada de um ponto sobre a curva é representada como uma função de um ou mais parâmetros. Para a equação com apenas um parâmetro, a posição vetorial de um ponto sobre a curva é fixada pelo valor do parâmetro. Se o parâmetro é t e a curva é bidimensional, pode-se escrever:

$X = F(t)$

$Y = G(t)$

A posição vetorial de um ponto sobre a curva é então dada por:

$$P(t) = (F(t), G(t))$$

Desde que um ponto sobre uma curva parametrizada é especificado por um único valor do parâmetro, a forma paramétrica é independente de eixos. A distância entre os pontos que descrevem a curva é fixada pela variação do parâmetro. É frequentemente conveniente normalizar o parâmetro de forma que a curva paramétrica caia entre $\emptyset \leqslant t \leqslant 1.0$.

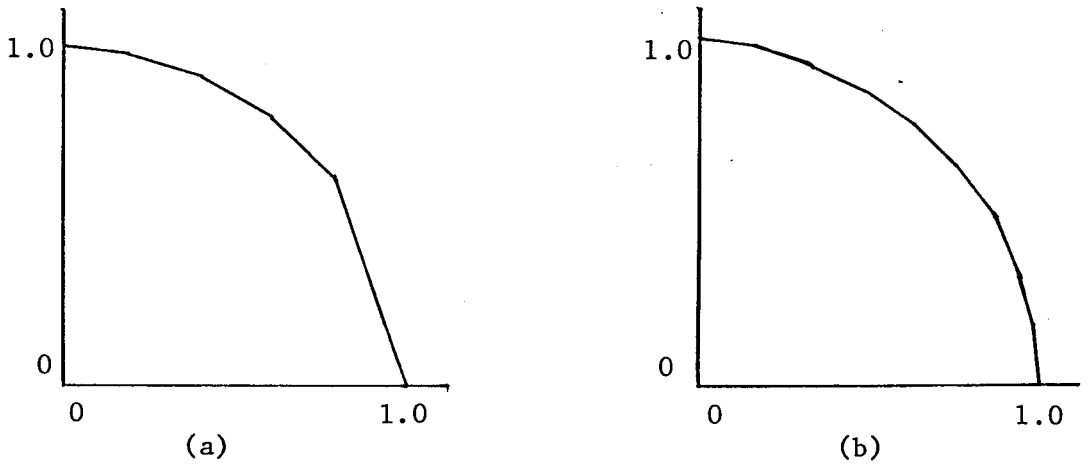



FIG: 5.5 - Representações de um Arco de Círculo

Uma comparação entre uma forma paramétrica e uma não paramétrica de um arco de círculo é mostrada na figura 5.5. Esta figura mostra que a escolha de funções paramétricas para curvas melhora a distribuição de pontos sobre uma curva, obtendo-se uma representação gráfica melhor. O quadrante de círculo mostrado na figura 5.5a é a representação do forma não paramétrica $Y=1-X^2$. Incrementos iguais de X foram utilizados para obter os pontos do arco, tendo como resultado comprimentos de arcos desiguais. A figura 5.5b, é obtida da forma parâmetrica básica dada por:

$$P(t) = (\cos\theta , \operatorname{sen}\theta)$$

Esta produz uma boa representação gráfica, desde que os comprimentos dos arcos obtidos são iguais. De modo geral, os pontos que descrevem um círculo podem ser obtidos pelas equações:

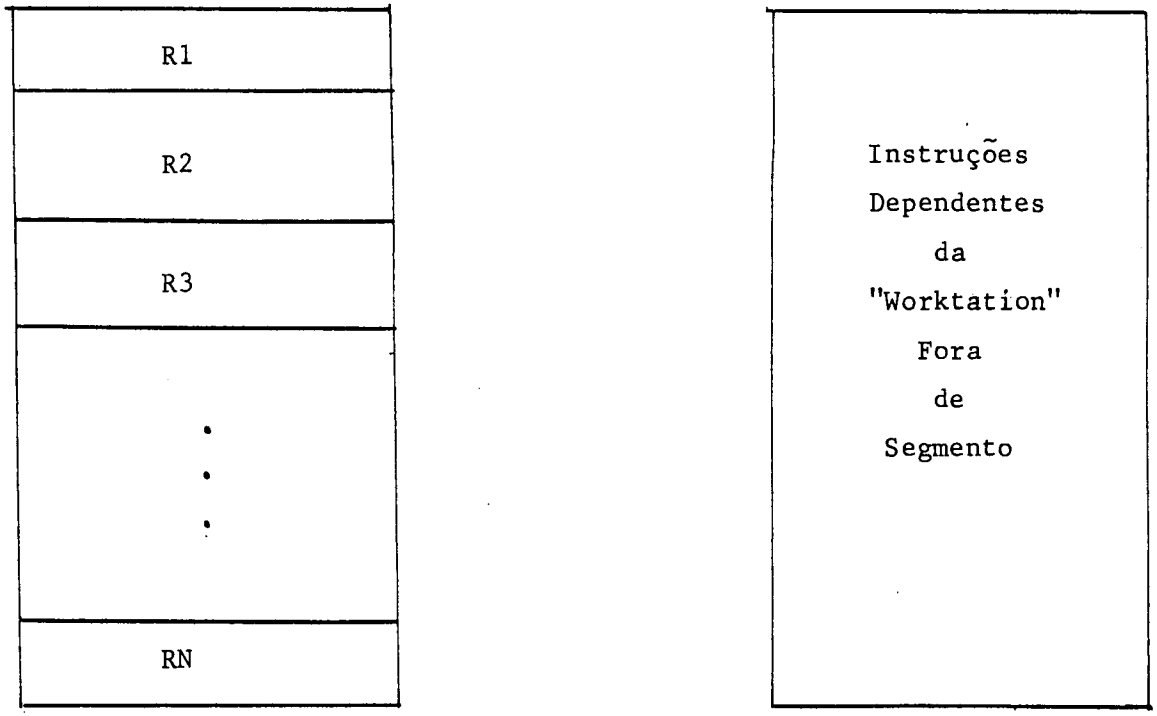
$$XN = R*COS(\theta + d\theta) + XC$$

$$YN = R*SIN(\theta + d\theta) + YC$$

Os parâmetros XC e YC indicam o centro da curva, o parâmetro R indica o raio da curva, e, desde que os últimos três parâmetros são sempre constantes, a variação do parâmetro $d\theta$ determina a distância entre os pontos XN e YN gerados e, com isso, a qualidade da representação gráfica obtida. O passo de variação do parâmetro $d\theta$ é escolhido de acordo com o tamanho da curva. Estas equações foram utilizadas para simulação das primitivas especiais círculo e arco de círculo.

## 5.4 ESTRUTURA DO ARQUIVO GRÁFICO

Uma característica que a estrutura de qualquer arquivo gráfico deve considerar, particularmente se êle faz parte de um sistema interativo, é a velocidade de transferência do programa gráfico ao processador gráfico. De modo geral, quanto mais estruturado for o arquivo gráfico, tanto mais complexos precisam ser o "software" e o "hardware" que transferem informação ao processador gráfico. As características de "hardware" do dispositivo devem ser aproveitadas ao máximo e, baseando-nos nesta premissa chegamos à estrutura do arquivo gráfico mostrada na figura 5.6. Nesta estrutura R1,R2,..., e RN são"records" que representam o resultado final do processamento dos segmentos do programa de aplicação. Cada "record" contêm um grupo de instruções do dependentesdo dispositivo que descrevem a representação gráfica do objeto descrito por seu segmento associado no programa de aplicação, incluidas as devidas transformações e cortes ("clipping"). Fisicamente todos os segmentos estão representados por um conjunto de instruções dependente do dispositivo contíguas. Sua separação lógica é obtida utilizando-se o mapa de correlação, apresentado na seção 5.6. Os blocos que representam os segmentos estão mostrados na figura 5.6, com tamanhos distintos, pois fisicamente cada segmento esta representado por um número variável de instruções dependentes do dispositivo. As primitivas do programa de aplicação geradas fora de segmento são processadas pela interface e alocadas fisicamente num outro bloco, que está no forma de um conjunto de instruções dependentes do dispositivo contíguas.




$R_i$(i = 1, ...,N) : "Records" que representam o resultado final do processamento de segmentos do programa de aplicação.

FIG: 5.6 - Estrutura Lógica do Arquivo Gráfico

O processador gráfico realiza periodicamente a regeneração da figura apresentada na tela-começa pelo segmento R1, executa todas as instruções dos segmentos até o fim de RN, executa intruções fora de segmento, e retorna ao começo de R1. A divisão em blocos contíguos de informações para instruções dependentes da "workstation" dentro e fora de segmentos, facilita a execução de regeneração implícita e a estruturação do mapa de correlação.

## 5.5 ATRIBUTOS NA INTERFACE

Nesta seção são abordadas as características e as implementações dos atributos das instruções executáveis e de segmento. O modo de atribuição de características às instruções do terminal é o mesmo para todos os atributos e pode ser visto na figura 5.7. Nesta figura é mostrado que as instruções fornecidas pelo núcleo passam num primeiro passo de interpretação por um arquivo intermediário. Este arquivo é um reflexo de todas as instruções executáveis e atributos para os quais é necessário gerar instruções do dispositivo. No segundo passo, todos os atributos das primitivas existentes em "hardware" no dispositivo são anexadas ao arquivo intermediário, gerando o arquivo de instruções dependentes da "workstation" que descrevem a última instrução executável e seus atributos, pronto para ser enviada para o satélite.

### 5.5.1 Atributos das primitivas de saída

Os três atributos de saída implementados, como mostrado na tabela 4.1, são: identificador de "pick", número "pen" e número "text".

O atributo identificador de "pick" é aplicável a todas as instruções executáveis dentro de segmento. Uma tabela de valores do identificador de "pick" associa estes valores aos conjuntos correspondentes de instruções dependentes da "workstation", pelos primeiros endereços ocupados pelos conjuntos no arquivo gráfico. Esta tabela será descrita na seção 5.6.

O atributo número "text" é aplicavel somente a instrução executável texto. Serão permitidos apenas dois tipos de caracteres: padrão e itálico . A qualidade do texto será do tipo zero, ou seja, a orientação do texto é sempre horizontal, e o tamanho do caracter e seu espaçamento são definidos pelo"hardware". Implementou-se esta característica pela adição da instrução STATSA a todo conjunto de instruções dependentes da "workstation" geradas sempre que é solicitada a

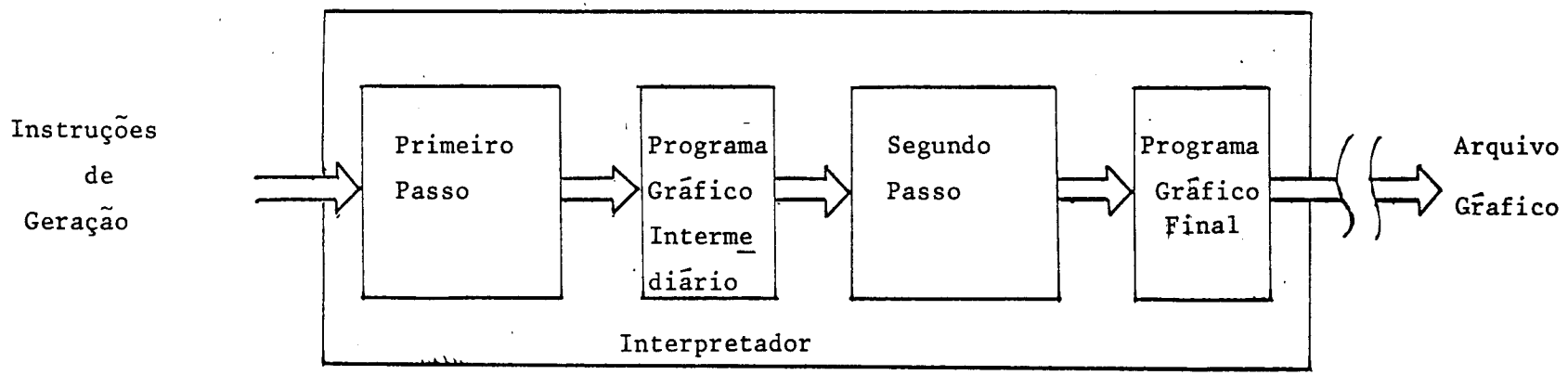



FIG: 5.7 - Estratégia de Agregamento de Atributos

execução da instrução executável texto, Uma tabela associando os valores dos números "text" aos endereço das instruções STATSA no arquivo gráfico, permite ao usuário a modificação destes valores, Esta tabela será descrita na seção 5.6.

| | Tipo de Linha | Intensidade |
|---|---|---|
| POLILINHA | X | X |
| POLIMARCADOR | - | X |
| GERAL | - | X |
| TEXTO | - | X |
| ÁREA | X | X |
| MATRIZ DE PIXELS | X | X |

Obs.: É permitido apenas uma largura de linha.

X = é válido - = não é válido

Tab. 5.3 - As instruções executáveis e as características de "pen".

O atributo número "pen" é associado a todas as instruções executáveis. A associação entre as características de "pen" e as instruções executáveis é mostrada na tabela 5.3. As instruções Área a Matriz de"Pixels", por serem simuladas por segmentos de reta como é mostrado nas seções 5.7.10 e 5.7.11, são afetadas pela característica tipo de linha.Implementou-se este atributo adicionando-se as características do número "pen" à primeira instrução dependente da "workstation" gerada após a definição deste índice. Uma tabela associado os valores dos números "pen" aos conjuntos correspondentes de instruções dependentes da "workstation", permite ao usuário modificar as características destes conjuntos no arquivo gráfico. Esta tabela será descrita na seção 5.6.

### 5.5.2 Atributos de segmento

Os três atributos de segmentos implementados,como pode ser observado na tabela 4.2, são: detetabilidade, piscamento e visibilidade.

Detetabilidade é um atributo de segmento e, em princípio, po

deria ser implementada apenas adicionando-se a característica de detetabilidade à primeira instrução dependente da "workstation" gerada dentro do segmento. Entretanto, a possibilidade de ter-se um conjunto de primitivas não detetáveis dentro do segmento obrigam-nos, quando associarmos a estas um valor do identificador de "pick" negativo, a fazer algumas considerações adicionais. A primeira instrução dependente da "workstation" gerada após a definição de um identificador de "pick" negativo, terá associada a si uma característica de não detetabilidade; após a esta associação, a primeira instrução dependente da "workstation" associada a um identificador de "pick" positivo, terá a característica de detetabilidade do segmento.

O atributo de piscamento é implementado adicionando-se a característica de piscamento à primeira instrução dependente da "workstation" gerada após a definição de um segmento.

O atributo de visibilidade é implementada de uma forma peculiar;através da inserção de uma instrução PULO no começo de cada segmento. Se o segmento é visivel o endereço de desvio é o da próxima instrução. Caso contrário, o endereço é o do inicio do próximo segmento. Implementou-se esta característica da seguinte forma ; desde que todas as instruções executáveis estão em coordenadas absolutas, qualquer expansão do interpretador sempre gera:

PONTO
X
Y
.
.
.

No caso de ser a primeira instrução dependente da "workstation" que se segue a definição de um segmento a sua expansão será:

PONTO
X
Y
PULO
ENDEREÇO
.
.
.

## 5.6 MAPA DE CORRELAÇÃO

O mapa de correlação é uma estrutura de dados que tem as seguintes propriedades:

- Prover o sistema com informações para processos interativos como, por exemplo, dado um endereço obtido pela atuação de um dispositivo de entrada da classe "pick", deve-se obter o identificador de "pick" e o nome do segmento associado.

- Prover o sistema com informações que permitam modificações dinâmicas de:

  . Atributos do segmento
  . Atributos das instruções executáveis
  . Remoção

Para a implementação das propriedades do mapa de correlação foram utilizadas as tabelas mostradas na figura 5.8. A seguir descrevemos as informações armazenadas nestas tabelas, e a utilização das mesmas.

A tabela de segmentos constitue-se dos seguintes dados:

APNOM - Apontador da tabela de segmento, aponta sempre para o nó do último nome de segmento que entrou na tabela.

NOM1 - Campo que contém os quatros primeiros caracteres do nome do segmento.

NOM2 - Campo que contém os quatro últimos caracteres do nome do segmento.

ENAPID - Campo que contém o índice do nó da tabela de "pick" onde se encontra o primeiro identificador de "pick" deste segmento.

A tabela de "pick" constitue-se dos seguintes dados:

APID - Apontador da tabela de "pick", aponta sempre para o nó do último valor do identificador de "pick" que entrou na tabela.

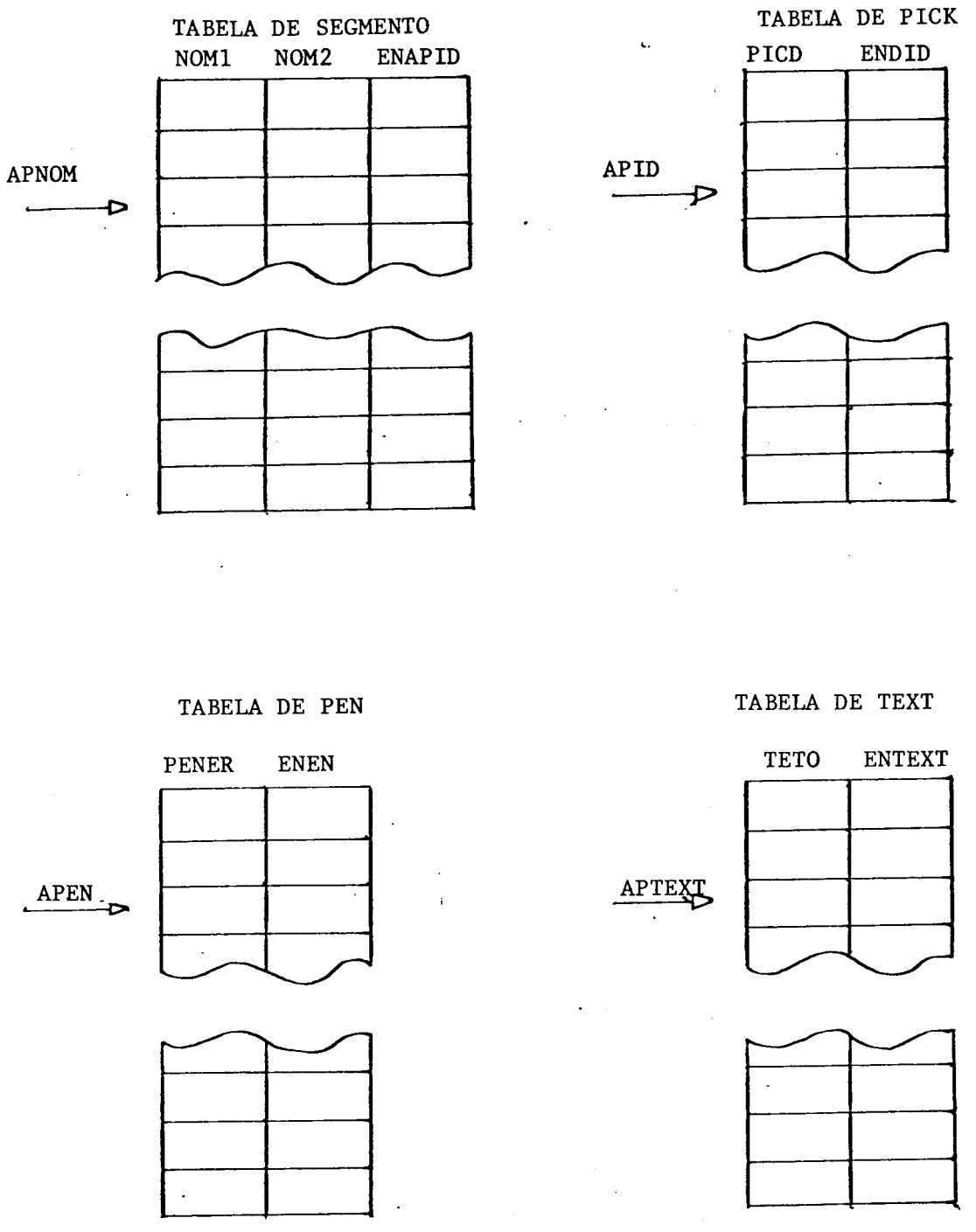



FIG: 5.8 - Estrutura Lógica do Mapa de Correlação

PICD - Campo que contém o valor do identificador de "pick" associado a um conjunto de instruções dependentes da "workstation" no arquivo gráfico, cujo endereço inicial está no campo ENDID do mesmo nó.

ENDID - Campo que contém o primeiro endereço do conjunto de instruções dependentes da "workstation" associadas ao valor do identificador de "pick" que esta no campo PICD do mesmo nó.

A tabela de "pen" constitue-se dos seguintes dados:

APEN - Apontador da tabela de "pen", aponta sempre para o nó do último valor do número "pen" que entrou na tabela.

PENER - Campo que contém o valor do número "pen" associado a um conjunto de instruções dependentes da "workstation" que utilizam suas características.

ENEN - Campo que contém o primeiro endereço do conjunto de instruções dependentes da "workstation" associadas ao valor do número de "pen" que esta no campo PENER do mesmo nó.

A tabela de "text" constitue-se dos seguintes dados:

APTEXT - Apontador da tabela de "text", aponta sempre para o nó do último valor do número "text" que entrou na tabela.

TETO - Campo que contém o valor do número "text" associado ao conjunto de isntruções dependentes da "workstation" que utilizam suas características.

ENTEXT - Campo que contém o endereço da primeira instrução STATSA do conjunto de instruções dependentes da "workstation" associadas ao valor do número "Text " que esta no campo TETO do mesmo nó.

Para solucionar o exemplo de provimento de informação para processos interativos, é facil observar que, como o identificador de "pick" está diretamente ligado ao endereço inicial do conjunto correspondente de instruções dependentes da "workstation", dado um endereço encontramos o identificador de "pick" associado. A partir do identificador de "pick" encontrado , o índice do nó onde ele se encontra é utilizado para encontrar o nome do segmento a ele associado.

Para modificações dinâmicas é interessante apresentar as soluções para cada um dos casos, conforme a divisão já apresentada.

A modificação dinâmica dos atributos do segmento pode ser subdividida em:

. Mudança de visibilidade

. Mudança de piscamento

. Mudança de detetabilidade

Para mudança de visibilidade do segmento, precisamos do endereço da instrução PULO associada. Para isto, devemos conseguir o endereço inicial do segmento que é facilmente obtido utilizando-se as tabelas de segmento e de "pick".

Para mudança de piscamento, precisamos do endereço inicial do segmento, também obtido das tabelas de segmento e de "pick".

Para mudança da detetabilidade, precisamos do endereço inicial do segmento e averiguarmos se existem identificadores de "pick" negativo dentro do segmento. Isto também é possível utilizando-se as tabelas de segmento e de "pick".

A mudança dinâmica dos atributos das instruções executáveis pode ser subdividida em:

. Mudança de "pen"

. Mudança de "text"

Para a mudança de "pen", precisa-se do endereço inicial do conjunto de instruções dependentes da "workstation" associado ao valor do número

"pen". Isto é possível utilizando-se a tabela de "pen".

A mudança de "text", precisa-se do endereço inicial da instrução STATSA do conjunto de instruções dependentes da "workstation" associado ao valor do número "text". Isto é possível utilizando-se a tabela de "text".

Para remoção das informações de um segmento na interface precisamos do endereço inicial e final do mesmo. Isto é possível utilizando-se tabelas de segmento e de "pick".

Como acabou de ser visto a estrutura do mapa de correlação é completa, pois atende a todas as necessidades que ela se propõe a atender.

## 5.7 ROTINAS DO INTERPRETADOR

Nesta seção abordam-se sucintamente as principais ações realizadas pelas rotinas do interpretador. Todas as instruções de geração estão disponíveis num arquivo sequêncial terminadas por um comando delimitado especial.Todas as instruções que a interface não tem capacidade para processar são analisadas e rejeitadas. Existe uma rotina que realiza a leitura do arquivo sequêncial e faz o encaminhamento de cada instrução de geração.

### 5.7.1 Abre segmento

A instrução Abre Segmento realiza basicamente as seguintes ações:

- Indica que as próximas instruções estão dentro de segmento
- Inicializa valores "default" dos atributos "pen","text" e identificador de "pick" das instruções executáveis.
- Inicializa valores "default" dos atributos de detetabilidade, piscamento e visibilidade do segmento.
- Inicializa a tabela de segmento no mapa de correlação, armazenando o nome do segmento.

### 5.7.2 Número "Pen"

A instrução Número "Pen" realiza basicamente as seguintes ações:

- Aciona uma variável lógica indicando que ocorreu um novo valor do número "pen".
- Coloca este novo valor em uma outra variável.

Estas variáveis são utilizadas por rotinas no segundo passo de atribuição de características (vide seção 5.5), para anexar as características associadas ao número "pen" ao primeiro conjunto de instruções dependentes da "workstation" geradas após esta definição. Caso este conjunto esteja definido dentro de um segmento, estas rotinas o associam a tabela de "pen".

### 5.7.3 Número"Text"

A instrução Número Text realiza basicamente as seguintes ações:

- Aciona uma variável lógica indicando que ocorreu um novo valor do número "text".
- Coloca este novo valor em uma outra variável.

Estas variáveis são utilizadas por rotinas no segundo passo de atribuição de características (vide seção 5.5), para anexar as características associadas ao número "text" ao primeiro conjunto de instruções dependentes da "workstation" geradas após esta definição e que as aceite. Caso este conjunto esteja definido dentro de um segmento, estas rotinas o associam a tabela de "text".

### 5.7.4 Identificador de "Pick"

A instrução Identificador de "Pick" realiza basicamente as seguintes ações:

- Aciona uma variável lógica indicando que ocorreu um novo identificador de "Pick".
- Coloca este novo valor em uma outra variável.

Esta variaveis são utilizadas por rotinas no segundo passo de atribuição de características (vide seção 5.5), para associar o primeiro conjunto de instruções dependentes da "workstation" geradas após esta definição à tabela de "pick".

### 5.7.5 Fecha Segmento

A instrução Fecha Segmento realiza basicamente a ação de indicar que as próximas instruções estão fora de segmento.

### 5.7.6 Polilinha

A execução da Polilinha ocorre em dois estágios: no primeiro, é aplicada uma transformação da "workstation" aos dados da instrução, gerando-se um arquivo com os pontos que descrevem a sequência dos segmentos de reta em coordenadas do dispositivo; no segundo, é gerado um programa gráfico otimizando, onde é minimizado o número de instruções dependentes da "workstation". O algoritmo básico utilizado pelo segundo estágio é mostrado na figura 5.9.

### 5.7.7 Polimarcador

A instrução Polimarcador é simulada pela utilização de caracteres. Cinco tipos de marcadores foram definidos e são mostrados na tabela 5.4. A instrução Polimarcador tem um tamanho único, definido pelo tamanho de caracter existente em "hardware".

| Tipo | Marcador |
|---|---|
| 1 | . |
| 2 | + |
| 3 | * |
| 4 | 0 |
| 5 | X |

Tab. 5.4 - Tipos de Marcadores

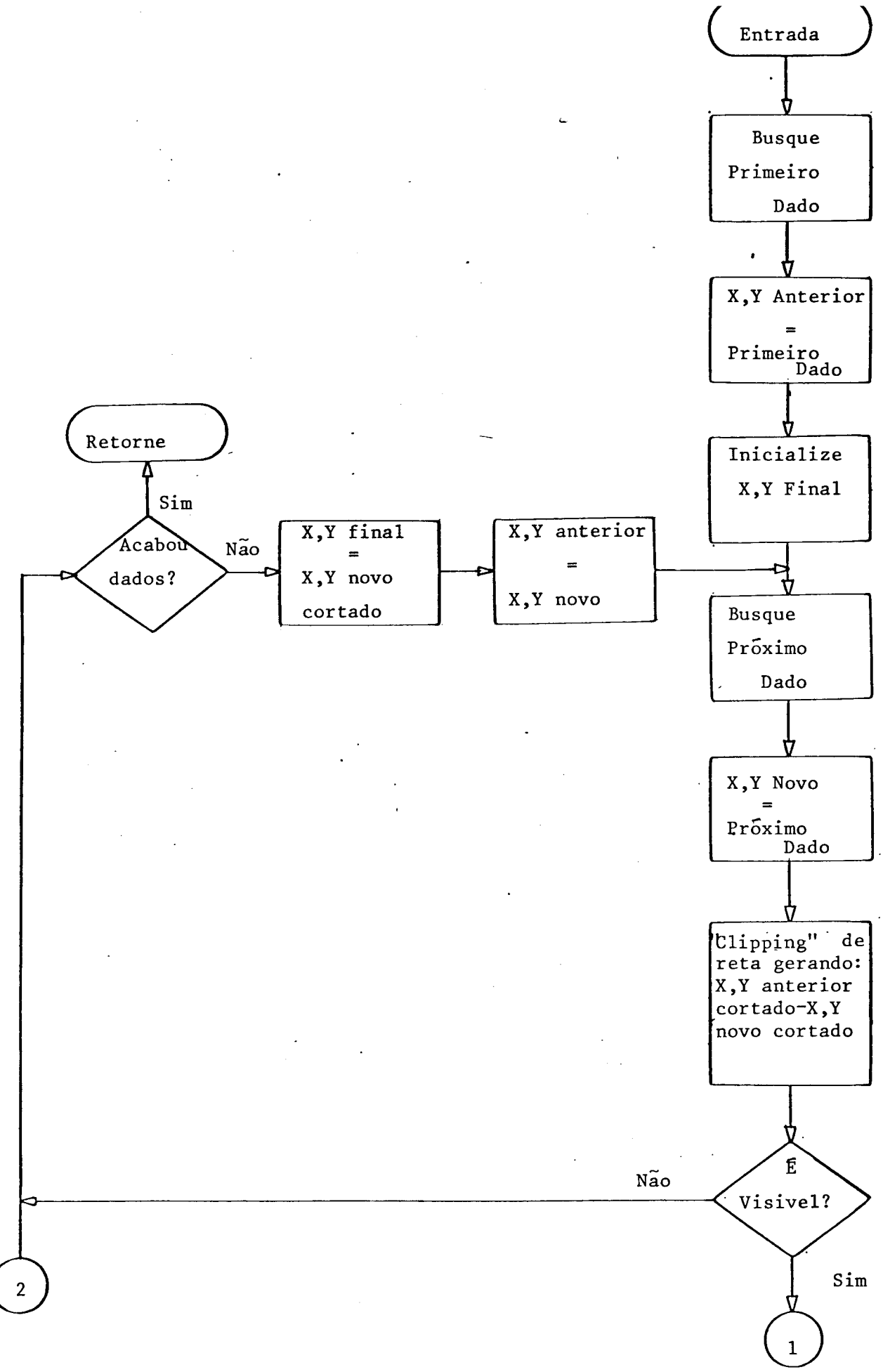



FIG: 5.9 - Algoritmo Básico para Geração Otimizada de Segmentos de Reta (continua)

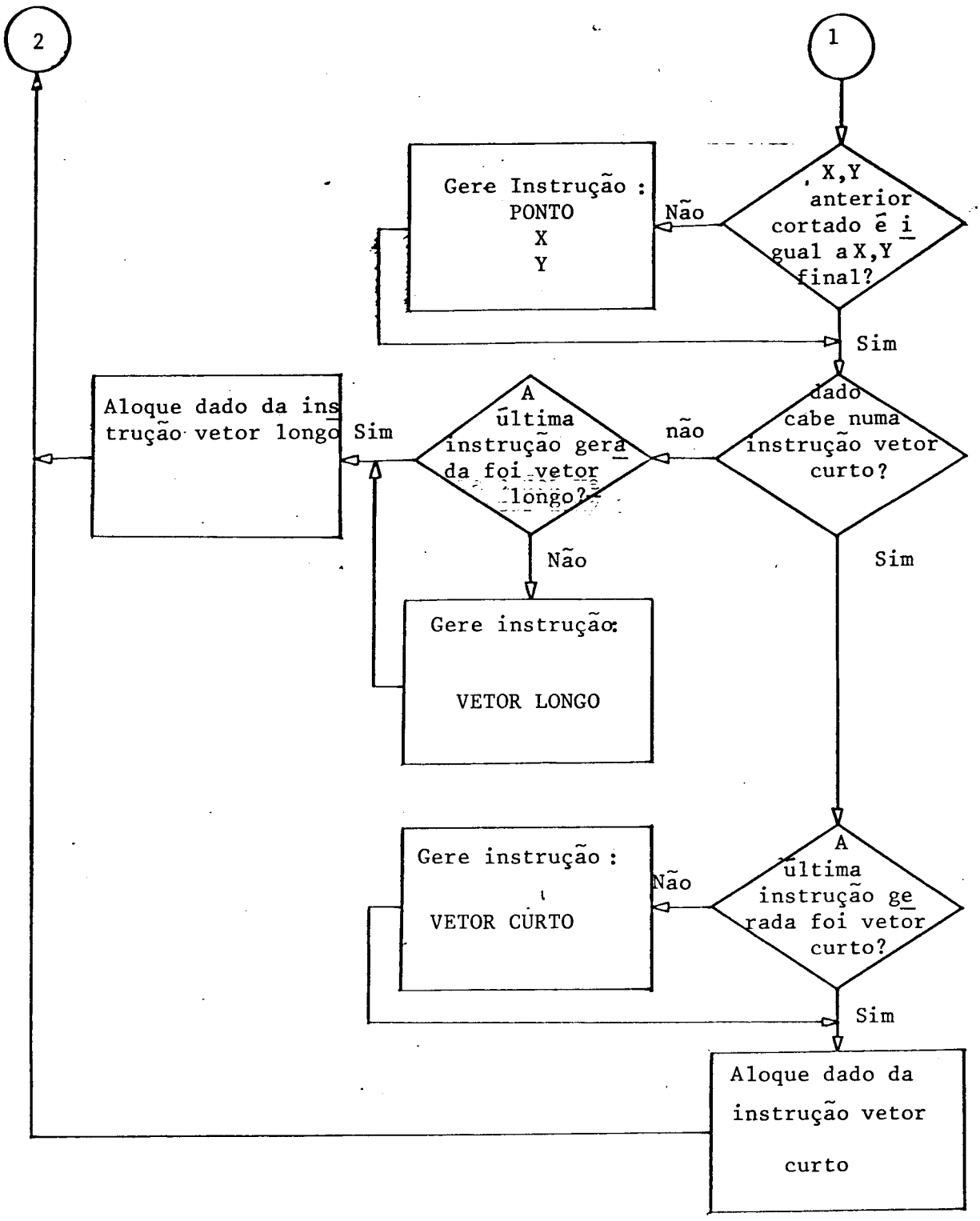



FIG: 5.5 - Algoritmo Básico para Geração Otimizada de Segmentos de Reta (continuação)

O expansor existente nesta rotina gera a macro mostrada abaixo para todo marcador completamente definido dentro do "viewport".

PONTO
X
Y
CARACTER
TIPO

### 5.7.8 Geral

A instrução Geral nesta implementação permite dois tipos de primitivas, círculo e arco de círculo, que são identificados pelos números 2 e 3, respectivamente. Neste caso, também existe um processo de dois estágios para a realização destas curvas. No primeiro, cria-se um arquivo em coordenadas do dispositivo; no segundo, a rotina do segundo estágio apresenta no item 5.7.6, gera um programa gráfico otimizado. As rotinas que realizam o primeiro passo, tiveram seus fundamentos apresentados na seção 5.3.3.

### 5.7.9 Texto

A instrução Texto é realizada pela utilização do gerador de caracteres existentes em "hardware" no terminal. A sequência de caracteres fornecidas por esta instrução, definidos dentro dos limites do "viewport", é expandida em instruções dependentes da "workstation" para:

PONTO
X
Y
STATSA
CARACTER
DADOS

Os dados que acompanham a instrução CARACTER estão compactadas numa forma especificada pelo "hardware" do terminal, que permite dois caracteres

por palavra .

### 5.7.10 Área

A instrução Área é realizada segundo normas de simulação para dispositivos vetoriais apresentadas no manual de definição do GKS / 6 /, onde se define que um dispositivo vetorial deve atender a instrução Área com, no mínimo,a representação do contorno do poligono especificado por esta. Desde que a sintaxe dos dados da instrução Área é similar à da instrução Polilinha, foi utilizado basicamente a rotina da instrução Polilinha (vide seção 5.7.6).

### 5.7.11 Matriz de "Pixels"

A instrução Matriz de"Pixels" é realizada segundo normas de simulação para dispositivos vetoriais apresentadas no manual de definição do GKS / 6 /, onde se define que um dispositivo vetorial deve atender a instrução matriz de "pixels" com,no mínimo, a representação do contorno das células especificadas dentro dos limites do "viewport". Para tanto, também existe um processo de execução em dois estágios: no primeiro, é criado um arquivo em coordenadas do dispositivo, descrevendo o contorno das células; no segundo, a rotina do segundo estágio apresentada na seção 5.7.6., gera um programa gráfico otimizado.

## 5.8 ROTINAS DO MODIFICADOR

Nesta seção abordam-se sucintamente as principais ações realizadas pelas rotinas do modificador.

### 5.8.1 Inicialização

A função Inicialização executa básicamente as seguintes ações:

- Inicialização das variáveis do mapa de correlação
- Inicialização das variáveis do GT-4Ø
- Inicialização de variáveis internas do interpretador
- Inicialização do arquivo gráfico

### 5.8.2 Atualização

A função Atualização executa básicamente a ação de regeneração implícita.

A função de Atualização visa atender a interfaces que permitam adiamento (vide seção 3.7). O tratamento dado ao adiamento nesta implementação foi o de atender apenas aos estados 1 ou 4, ou seja, toda ação do programa de aplicação deve ser realizada o mais rapido possível, e a única ação que é possível adiar é a da regeneração implícita da superfície de visualização.

### 5.8.3 Remoção

A função Remoção executa básicamente as seguintes ações:

- Retira do mapa de correlação todas as informações do segmento removido.
- Atualiza todos os dados dos outros segmentos.
- Retira do arquivo gráfico o segmento a ser removido.

No mapa de correlação faz-se uma compactação e atualização dos dados das tabelas abaixo dos nós removidos. A remoção de segmentos do arquivo gráfico foi implementadas de forma a não existir necessidade de parar o processador gráfico. Para isso, foi utilizado um esquema de retirada temporária e compactação de segmentos no arquivo gráfico,ilustrado na figura 5.10. Na figura 5.10a temos o bloco original de segmentos de onde se deseja remover o "record" R2. Para realizarmos isso, apontamos a instrução PULO do "record" R2 para o "record" seguinte ao próximo "record" a ser compactado, R4, como está mostrado na figura 5.10b. Ao compactarmos R3 copiamos também o começo do próximo record a ser compactado,e apontamos a última instrução PULO para o "record" seguinte ao próximo "record" a ser compactado, R5, como esta mostrado na figura 5.10c, continuando este processo até a compactação do "record" RN.

### 5.8.4 Mudança de Nome

A função Mudança de Nome executa a ação de substituir o nome

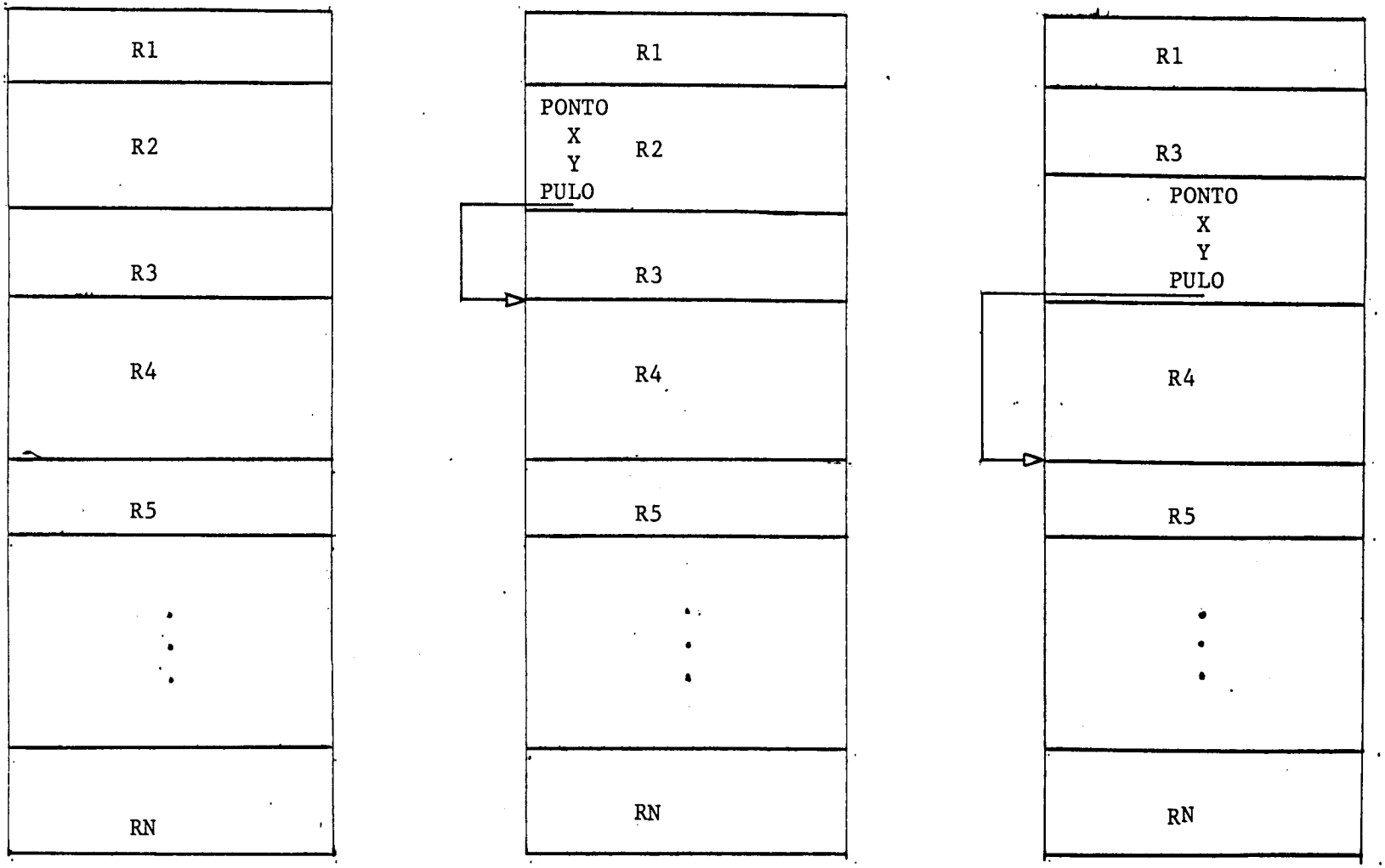



FIG: 5.10 - Princípio Utilizado para Remoção de Segmento no Arquivo Gráfico

antigo do segmento pelo novo na tabela de segmento.

### 5.8.5 Mudança de Visibilidade

A função Mudança de Visibilidade executa básicamente as seguintes ações:

- Determina o endereço no arquivo gráfico da instrução PULO do segmento e para onde este deve apontar, utilizando as tabelas de segmento e de "pick",
- Modifica a característica de visibilidade no arquivo gráfico.

### 5.8.6 Mudança de Detetabilidade

A função Mudança de Detetabilidade executa básicamente as seguintes ações:

- Determina os endereços das instruções no arquivo gráfico que mudaram suas características de detetabilidade, utilizando as tabelas de segmento e de "pick".
- Modifica a característica de detetabilidade destas instruções no arquivo gráfico.

### 5.8.7 Mudança de Piscamento

A função Mudança de Piscamento executa básicamente as seguintes ações:

- Determina o endereço da primeira instrução dependente da "workstation" do segmento, utilizando as tabelas de segmento e de "pick".
- Modifica a característica de piscamento desta instrução no arquivo gráfico.

### 5.8.8 Mudança de "Pen"

A função Mudança de "Pen" executa básicamente as seguintes ações:

- Determina todos os endereços dos conjuntos de instruções no arquivo gráfico associados ao valor do número "pen" na tabela de "pen".
- Modifica as características de "pen" destes conjuntos no arquivo gráfico.

5.8.9 Mudança de "Text"

A função Mudança de "Text" executa básicamente as seguinte ações:

- Determina todos os endereços dos conjuntos de instruções no arquivo gráfico associados ao valor do número "text" na tabela de "text".
- Modifica a característica de "text" destes conjuntos no arquivo gráfico.

# CAPÍTULO 6

# CONSIDERAÇÕES FINAIS

Para concluir este trabalho apresentamos uma discussão a respeito dos objetivos ou benefícios atingidos com o desenvolvimento deste trabalho, além de fazermos menção a trabalhos que dão prosseguimento a esta tese.

Dois resultados significativos foram obtidos neste trabalho:

- A especificação explícita, no sistema gráfico de base, da separação entre as partes independente e dependente relacionadas com dispositivos gráficos de saída.

- A definição de uma interface padronizada para todos os tipos de dispositivos gráficos de saída.

Estes resultados, apesar de não resolverem todos os problemas relativos aos dispositivos gráficos de saída, podem contribuir para uma utilização mais ampla de sistemas gráficos pela eliminação de suas diferenças estruturais. Existe ainda uma série de problemas relativos a dispositivos gráficos de saída que impedem a obtenção de uma independência total; as diferenças funcionais entre dispositivos é a principal. Por exemplo, a diferença de precisão entre dispositivos gráficos de saída pode nos levar a têrmos uma reta com graus de distorção diferentes em dispositivos distintos. As próprias dificuldades encontradas na implementação da interface para um vídeo vetorial regenerativo, tanto na própria adaptação dos recursos do dispositivo quanto nos aspectos da eficiência desta adaptação às capacidades previstas pelo núcleo, reiteram a dificuldade de alcançar-se uma independência total . Contudo, nota-se que os esforços realizados ao longo dos últimos anos vêm paulatinamente alcançando melhores resultados no sentido de obtenção de um maior grau de independência.

A interface implementada foi submetida apenas a testes corretivos, ou seja, verificou-se somente se as rotinas que a compõem funcionavam corretamente. O teste integrado da interface somente se dará quando dispusermos de um sistema gráfico todo implementado. A avaliação de características como tempo de resposta e facilidade de utilização pelo usuário levar-nos-á a conclusões mais definitivas sobre a eficiência do sistema como um todo.

Para o prosseguimento do trabalho iniciado por esta tese, listamos alguns tópicos enquadrados dentro do objetivo de implantar-se um sistema de processamento de informação gráfica para controle de processos:

- Implementação de uma interface para um banco de dados.
- Implementação do núcleo gráfico.
- Implementação de uma interface do usuário para aplicações em controle de processos.

Nesta lista não está citado a implementação da interface de entrada, trabalho realizado em paralelo ao desta tese e que deverá ser brevemente apresentado /8/. Os trabalhos sugeridos são objeto de uma tese de doutoramento /14/ e dão sequência a uma linha de trabalho adotada dentro da FEC.

## BIBLIOGRAFIA


/ 1 / GILOI,W.K. : "Interactive Computer Graphics" - Prentice Hall - 1978

/ 2 / TOZZI, C. _ "Um Terminal Gráfico Interativo Baseado na Tecnologia do TV-Raster" - Tese de Doutoramento - UNICAMP - 1979.

/ 3 / FOLEY, J.D. e WALLACE,V.L. ; "The Art of Natural Graphic Man-Machine Conversation" - Proceedings IEEE - Abril 1974.

/ 4 / "Status report of the Graphics Standards Planning Committe of ACM/SIGRAPH" 1978.

/ 5 / "Methodology in Computer Graphics" - Editores Guedj, R.A. e Tucker, H.A. - North-Holland - 1979.

/ 6 / "Information Processing Graphical Kernel System (GKS), Funcional Description" versão 6.2.

/ 7 / DALTRINI,B.M.; JINO M. e MESSINA, L.A. : "Um sistema Integrado Núcleo Gráfico/Banco de Dados para tratamento da Informação"- II Simpósio sobre aplicações gráficas por computador e sistemas gráficos interativos (SUCESU - agosto 1980).

/ 8 / MARTINS,R.P. : "Interface de Entrada um Núcleo Gráfico" - Tese de Mestrado- a ser apresentada.

/ 9 / NEWMAN, W.M. e SPROULL,R.F. ; "Principles of Interactive Computer Graphics", 2nd Edition - McGRAW-Hill - 1979.

/10/ KILGOUR,A.C. : "The Evaluation of a Graphics System for Linked Computers "- Software Pratice and Experience - Vol. 1, 1971.

/11/ DALTRINI, B.M.; MARTINS,R.P.; MOLINARO, L.F.R.; JINO,M. e NETTO ANDRADE,M.L.: "Graphical Kernel System in an Information System for Process Control"- International Conference on CAD/CAM as a Basis for the Development of Technology in Developing Nations - outubro 1981.

/12/ "GT-40 User's Guide" - DEC-11-HGTGA-A-D- Digital Equipment Corpotation - 1973.

/13/ "Processor hadbook PDP-11/05" - Digital Equipment Corporation - 1973

/14/ DALTRINI,B.M. - Tese de Doutoramento - em andamento.