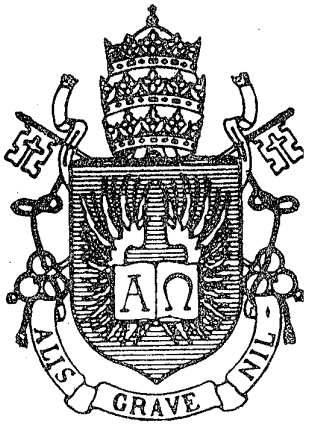


PUC

Série: Monografias em Ciência da Computação
Nº 3/84

PADRONIZAÇÃO DE SISTEMAS GRÁFICOS

por

Cláudia Alvarenga
Astério K. Tanaka
Rubens N. Melo

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453
RIO DE JANEIRO – BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, Nº 3/84
Editor: Antonio L. Furtado Agosto, 1984

PADRONIZAÇÃO DE SISTEMAS GRÁFICOS *

por

Cláudia Alvarenga
Astério K. Tanaka**
Rubens N. Melo

* Trabalho parcialmente financiado pela FINEP.
** Instituto Militar de Engenharia

RESUMO

Muito esforço tem sido investido na última década visando um ade - quado mínimo para um pacote gráfico. Este trabalho apresenta as duas principais propostas para a padronização de tais softwares: o sistema Core e o sistema GKS. Primeiramente estes sistemas são analizados e depois comparados nas suas principais características.

PALAVRAS CHAVE

Computação Gráfica, Processamento Gráfico, Pacotes Gráficos, CORE, GKS, Padronização.

ABSTRACT

Much effort has been invested during the last decade towards an adequate minimum for a graphics package.This work presents the two main proposals for the standardization of such softwares: the CORE system and the Graphical Kernel System (GKS). First these systems are analysed and then compared in their main features.

KEY WORDS

Computer Graphics, Graphic processing, Graphics packages,CORE,GKS, Graphics standards.

S U M Á R I O

1. INTRODUÇÃO

Logo que surgiram os primeiros trabalhos em Computação Gráfica, em 1963, ficou clara a sua importância, já que ela possibilita o uso de informação gráfica na comunicação com sistemas de computação. Porém, inicialmente, o equipamento era caro e as técnicas para a sua programação não eram satisfatórias. Como todas as outras áreas de sistemas digitais, a Computação Gráfica tem sido bastante afetada com a revolução da Microeletrônica que, com a crescente miniaturização dos circuitos, está possibilitando o aumento de potência e uma redução considerável nos custos desses sistemas.

A disponibilidade de qualquer tecnologia nova de computação é sempre seguida pelo desenvolvimento de práticas comuns para seu uso, geralmente aceitas pela comunidade da área. Dessa forma, o "software" torna-se mais portável, o "hardware" pode ser mais compatível e as pessoas envolvidas começam a se entender. Mas essas práticas informais não se mostraram suficientes na área de Computação Gráfica devido a problemas específicos como falta de disponibilidade de bons programadores gráficos, elevado preço de produção de "software" gráfico, tendência idiossincrática do "hardware", entre outros.

A partir de 1974, tornou-se evidente uma tendência de desenvolvimento de padrões gráficos, visando a unificação e o amadurecimento da área, o que acarretaria o seu progresso e consequente disseminação.

O presente trabalho procura apresentar e analisar os principais esforços de padronização de subrotinas gráficas básicas que surgiram desde então.

2. PADRÕES EM COMPUTAÇÃO

Padrões são vitais na nossa vida diária e como consumidores nos tornamos dependentes deles. Quando entendemos a sua importância no dia-a-dia, é mais fácil entender porque surgem necessidades de padronização em diversas áreas.

Organizações de padronização existem para promover o desenvolvimento e aprovação de padrões que representam um consenso.

2.1 PRINCIPAIS ORGANIZAÇÕES DE PADRONIZAÇÃO

No início do século, a necessidade de padrões era razoavelmente difundida e, consequentemente, foram fundadas várias organizações para desenvolvimento de padrões numa grande variedade de áreas. Por isso, quando surgiram as primeiras sugestões para estabelecimento de padrões na área de computadores, já haviam disponíveis organizações maduras e bem estabelecidas para aceitar essa responsabilidade.

A nível internacional, a "International Standards Organization" (ISO) autorizou a formação dos Comitês Técnicos TC97 (Computadores e Processamento de Informação) e TC95 (Máqui-

nas de Escritório), que depois se uniram no TC97 (Sistemas de Processamento de Informação). Hoje, além da ISO, as organizações internacionais mais conhecidas da comunidade de computação são a "International Electrotechnical Commission" (IEC) e o "Comité Consultatif International Télèphonique et Télègraphique" (CCITT) da "International Telecommunication Union" (ITU).

Nos E. U. A., o "American National Standards Institute" (ANSI) atribuiu à "Business Equipment Manufacturers Association" - hoje a "Computer and Business Equipment Manufacturers Association" (CBEMA) - a responsabilidade de criar os comitês de padrões nacionais americanos X3 e X4, que se uniram posteriormente no comitê ANSI/X3 (Sistemas de Processamento de Informação). O ANSI/X3 tem agora a responsabilidade de todos os padrões ANSI relacionados com computadores.

Na Europa, formou-se a "European Computer Manufacturers Association" (ECMA), em meados de 1961. Hoje, existem várias organizações nacionais de padronização na Europa, entre elas o "Deutches Institut für Norming" (DIN), da Alemanha Ocidental.

2.2 EXEMPLOS DE PADRÕES EM COMPUTAÇÃO

São vários os exemplos de padrões em computação. São descritos sucintamente, a seguir, alguns padrões importantes e conhecidos, classificados por área.

2.2.1 Bancos de Dados

Em abril de 1971, foi publicado pelo "Data Base Task Group"

(DBTG) da "Conference on Data Systems Languages" (CODASYL) um documento contendo a proposta de linguagens para a descrição de dados e uma linguagem para a manipulação de dados em um banco de dados. Essa proposta foi bastante criticada e em 1972 foi estabelecido pelo "Standard Planning and Requirements Committee" (SPARC) do ANSI-X3 o "ANSI/X3/SPARC Study Group on Data Base Management Systems", cujo objetivo era determinar as áreas, se houvessem, de tecnologia de bancos de dados para as quais a atividade de padronização fosse apropriada, e produzir um conjunto de recomendações em tais áreas. Esse grupo identificou o aspecto "interfaces" como o único de um sistema de banco de dados que poderia possivelmente ser padronizado, e definiu, então, uma arquitetura para esses sistemas e suas "interfaces", em 1975. Em 1978, a CODASYL publicou um relatório interno ajustando a sua proposta inicial de 1971 às recomendações do ANSI/X3/SPARC. Muitos sistemas de gerência de bancos de dados foram desenvolvidos orientados pelos conceitos estabelecidos por essas duas entidades.

2.2.2 Linguagens de Programação

Dois exemplos de linguagens padronizadas são bastante significativos, considerando-se a sua popularidade : COBOL e FORTRAN.

COBOL é a linguagem mais utilizada em aplicações computacionais comerciais desde o seu surgimento por volta de 1960. O órgão responsável pelo desenvolvimento do COBOL é o "CODASYL COBOL Committee", que submeteu a linguagem ao ANSI, que o aceitou como padrão em 1968. A última versão é de 1974, e o "COBOL

Journal of Development" publicado a cada 2 ou 3 anos serve como a especificação oficial da linguagem.

FORTRAN foi uma das primeiras - final da década de 50 - linguagens de programação de alto nível, e é hoje, provavelmente, a linguagem mais utilizada na programação científica e de engenharia. Inicialmente, não havia uma definição precisa da linguagem. As primeiras versões foram desenvolvidas especificamente para máquinas IBM; subsequentemente, foram produzidas versões de FORTRAN para outras máquinas. Nesse processo, foram introduzidas muitas variações significativas. Em 1966, o comitê X3J3 do ANSI publicou a proposta do que seria a definição da linguagem (independente de máquina). Esse padrão ainda evoluiu e foi publicada uma proposta revisada do X3J3 do ANSI em 1977, atualmente a proposta de padronização do FORTRAN utilizada.

2.2.3 Redes de Computadores

Em meados dos anos 70 surgiram as primeiras redes comerciais juntamente com a necessidade de se estabelecer uma "interface" padrão entre o computador e a rede. Visando facilitar o trabalho de interconexão de redes e permitir que fabricantes desenvolvessem "software" e "hardware" portáveis, o CCITT resolveu criar uma série de padrões para redes públicas comutadas por pacotes. Esses padrões são conhecidos como recomendações da série X. Em particular, a recomendação X.25 do CCITT descreve o protocolo padrão de acesso ou "interface" entre o computador e a rede.

3. PADRÕES EM COMPUTAÇÃO GRÁFICA

A grande variedade de dispositivos gráficos tem sido um problema básico em Computação Gráfica desde o seu início no começo dos anos 60. Os primeiros pesquisadores tinham muitas dificuldades em partilhar seus resultados. Com o passar do tempo, os pesquisadores começaram a se concentrar em classes de "hardware" gráfico, o que levou à formação de várias escolas de projeto de sistemas gráficos. A primeira divisão que surgiu foi entre usuários de "plotters" e usuários de "displays" interativos. O grupo de "displays" interativos trabalhava com "displays" do tipo "refresh", mas depois surgiram os "displays" do tipo "storage tube" e, recentemente, os "displays" do tipo "raster". O estudo e a pesquisa nessas escolas logo identificaram várias áreas semelhantes entre elas, tornando possível a consideração do desenvolvimento de padrões em Computação Gráfica.

3.1 PORTABILIDADE

A única justificativa forte para padronização em Computação Gráfica é o estímulo à portabilidade de programa, isto é, a capacidade de se transportar aplicação de uma instalação para outra com o mínimo de alterações no programa. Durante muito tempo, as dificuldades de transporte de programas de aplicação entre instalações foram proibitivas para muitos usuários gráficos potenciais. Outro problema é a necessidade de treinar programadores até entenderem as complexidades e idiossincrasias de cada

nova instalação gráfica que eles encontram; a padronização oferece a oportunidade de reduzir esse problema de treinamento e atingir a portabilidade de programador.

A padronização do "hardware" gráfico concorreria para a portabilidade, mas isso não ocorrerá num futuro próximo, e mesmo que ocorresse, não evitaria o desenvolvimento de enfoques diferentes para o uso do mesmo "hardware". Assim, esse problema só pode ser resolvido através do fornecimento de "interfaces" padronizadas entre o programa de aplicação e o dispositivo gráfico.

3.2 ATIVIDADES INICIAIS DE PADRONIZAÇÃO

A união gradual de diferentes escolas de projeto de sistemas gráficos e a crescente consciência da necessidade de portabilidade estabeleceram o início dos esforços na área de padronização.

Em abril de 1974, foi promovido pela ACM o "Workshop on Machine Independent Graphics" sob patrocínio do "National Bureau of Standards", em Garthersburg, Maryland, EUA. Naquele encontro, foram discutidas as regras básicas para o desenvolvimento de um futuro padrão para a Computação Gráfica. Seguiram-se outros encontros e trocas de artigos, mas o progresso era lento e poucos os resultados.

Apenas em maio de 1976 é que esses esforços foram revitalizados no "Workshop on Graphics Standards Methodology", promovido pela IFIP em Seillac, França. Esse encontro focalizou a atenção em alguns ítens fundamentais do projeto de sistemas gráficos, consequência da conscientização da necessidade de uma metodologia

na qual basear o projeto desses sistemas. O principal resultado desse encontro foi a identificação das vantagens da separação das funções essenciais para geração de figuras - o núcleo - de outras funções, como a modelagem, específicas de uma aplicação.

3.3 SISTEMAS DE MODELAGEM

O encontro de Seillac ressaltou a importância do relacionamento entre o projeto de sistemas gráficos e a função de modelagem. Os sistemas gráficos iniciais incluiam funções de modelagem mas, com o passar do tempo, a "interface" entre o programa de aplicação e o sistema gráfico foi deslocada de forma que a responsabilidade da modelagem fosse da aplicação.

Segue a reprodução do trecho do relatório de Seillac referente a esse tópico :

"A solução parece residir na divisão das facilidades necessárias em duas partes, o sistema gráfico e o sistema de modelagem. O sistema de modelagem permite que se definam objetos em suas próprias coordenadas; ele fornece funções para tranformações de composição e combinação, e para a aplicação dessas transformações em coordenadas locais. Os resultados dessas transformações produzem tipicamente uma definição da informação visualizada num sistema de coordenadas universal, próprio para aplicação da transformação de visualização. Através do sistema gráfico, pode-se estabelecer a transformação de visualização apropriada e gerar a figura usando a definição em coordenadas universais; isto será feito tipicamente pela chamada de funções de desenho de linhas

com argumentos em coordenadas universais".

Esse enfoque evita muitos dos problemas inerentes ao uso de um único conjunto de rotinas de transformação. Na fase de modelagem, cada transformação recebe um ponto em coordenadas locais e produz um ponto ainda em coordenadas locais (ou universais).

O sistema de modelagem tenderá a conter um conjunto básico de funções que incluem:

- função para composição de transformação;
- uma pilha de transformação;
- uma forma de manipular as transformações na pilha com operações de "push" e "pop";
- função para aplicar a transformação corrente num ponto em coordenadas locais.

Haverá necessidade, entretanto, de muitos tipos diferentes de sistemas de modelagem, para cada aplicação específica que envolve modelagem. Apesar dessa diversidade, pode-se esperar que todos os sistemas de modelagem contenham as funções acima listadas de uma forma ou de outra; além disso, é viável a sugestão de um sistema de modelagem básico como um ponto inicial para o desenvolvimento de sistemas mais dependentes de aplicação.

3.4 ESTRATÉGIAS DE PROJETO

Para se fornecer ao programador primitivas gráficas para controle de geração de figuras e manipulação de interação, existem três estratégias básicas: projeto de uma linguagem gráfica,

projeto de extensões gráficas para uma linguagem já existente e projeto de um pacote de subrotinas a serem embutidas em uma linguagem hospedeira já existente.

O projeto de uma nova linguagem é uma tarefa difícil. Os principais fatores que dificultam essa estratégia são a falta de consenso sobre as capacitações funcionais de suas primitivas, os objetivos da linguagem e a dificuldade do estabelecimento de credibilidade para qualquer linguagem nova.

A extensão de uma linguagem envolve aumentar a linguagem e, o mais importante, mudar o compilador. Para a maioria dos compiladores em produção, este é um processo complexo e caro, que acarretará um "overhead" potencial para todos os usuários, independente da utilização dessas facilidades.

Com um pacote de subrotinas, não existem alterações na linguagem nem no compilador, apenas no ambiente de execução. O desenvolvimento e a alteração num sistema desse tipo são mais simples que nos casos anteriores. Além disso, pode-se testar sintaxes e semânticas diferentes sem afetar outros usuários. Por outro lado, uma sintaxe que só permita listas de parâmetros não é interessante. Pacotes de alto nível serão penalizados em espaço e tempo de execução devido ao ambiente de execução maior.

Como as vantagens sobrepujaram as desvantagens, o mecanismo de pacote de subrotinas foi a estratégia aceita. O objetivo de um pacote de subrotinas é ser utilizável em linguagens de alto nível, com uma interface para o usuário que é independente de

dispositivo, máquina e sistema operacional.

3.5 INDEPENDÊNCIA DE DISPOSITIVO

A estratégia fundamental para se atingir a portabilidade é a realização do conceito de independência de dispositivo. O significado desse conceito é mostrado na figura abaixo.

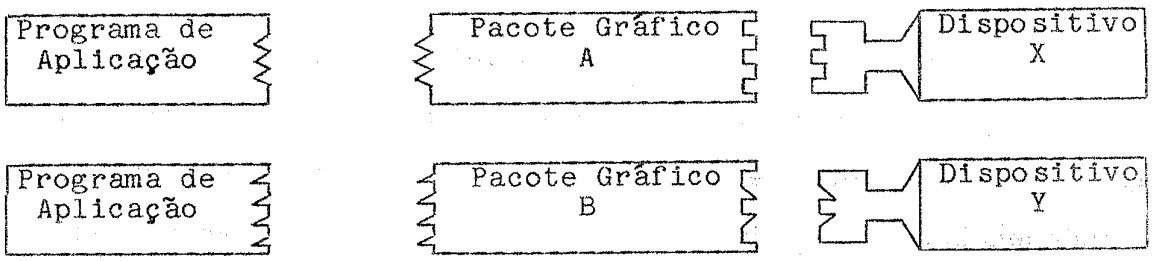


figura (a)

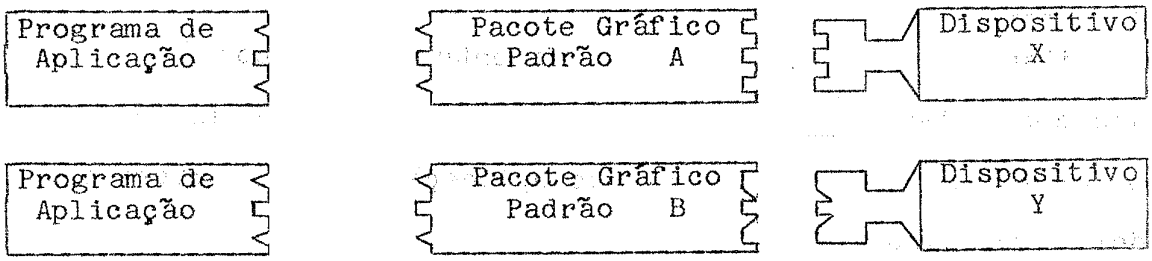


figura (b)

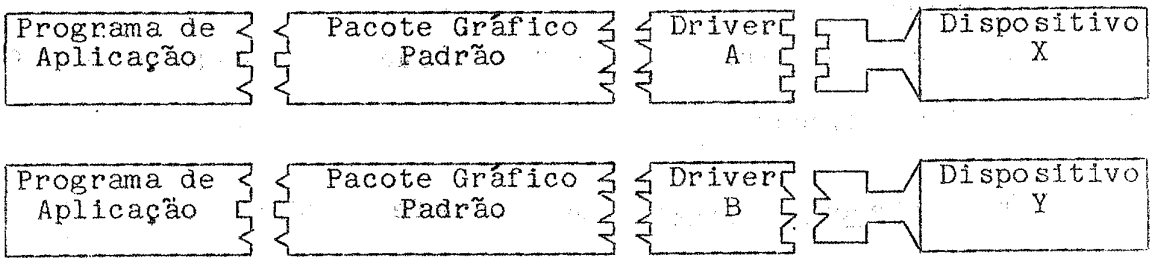


figura (c)

A primeira figura (a) mostra dois pacotes gráficos desenvolvidos para dois "displays" diferentes; cada um apresenta uma "interface" de programação diferente para os programas de aplicação, que são consequentemente diferentes. Esses pacotes são chamados dependentes de dispositivo porque forçam o programador de aplicação a ajustar o seu programa às facilidades específicas do dispositivo.

Na figura (b), é mostrada a tentativa de se fornecer uma "interface" uniforme para o programador; o mesmo programa de aplicação pode ser executado em qualquer instalação. Atingiu-se a independência de dispositivo para o programa de aplicação, mas os dois pacotes gráficos ainda dependem do dispositivo.

Na figura (c), a única parte dependente de dispositivo é o "driver" de dispositivo. Essa "interface" uniforme permite que os "displays" e os dispositivos de entrada gráficos sejam tratados como dispositivos lógicos, cujas semelhanças são abstraídas para benefício do programador de aplicação e cujas diferenças são dele escondidas.

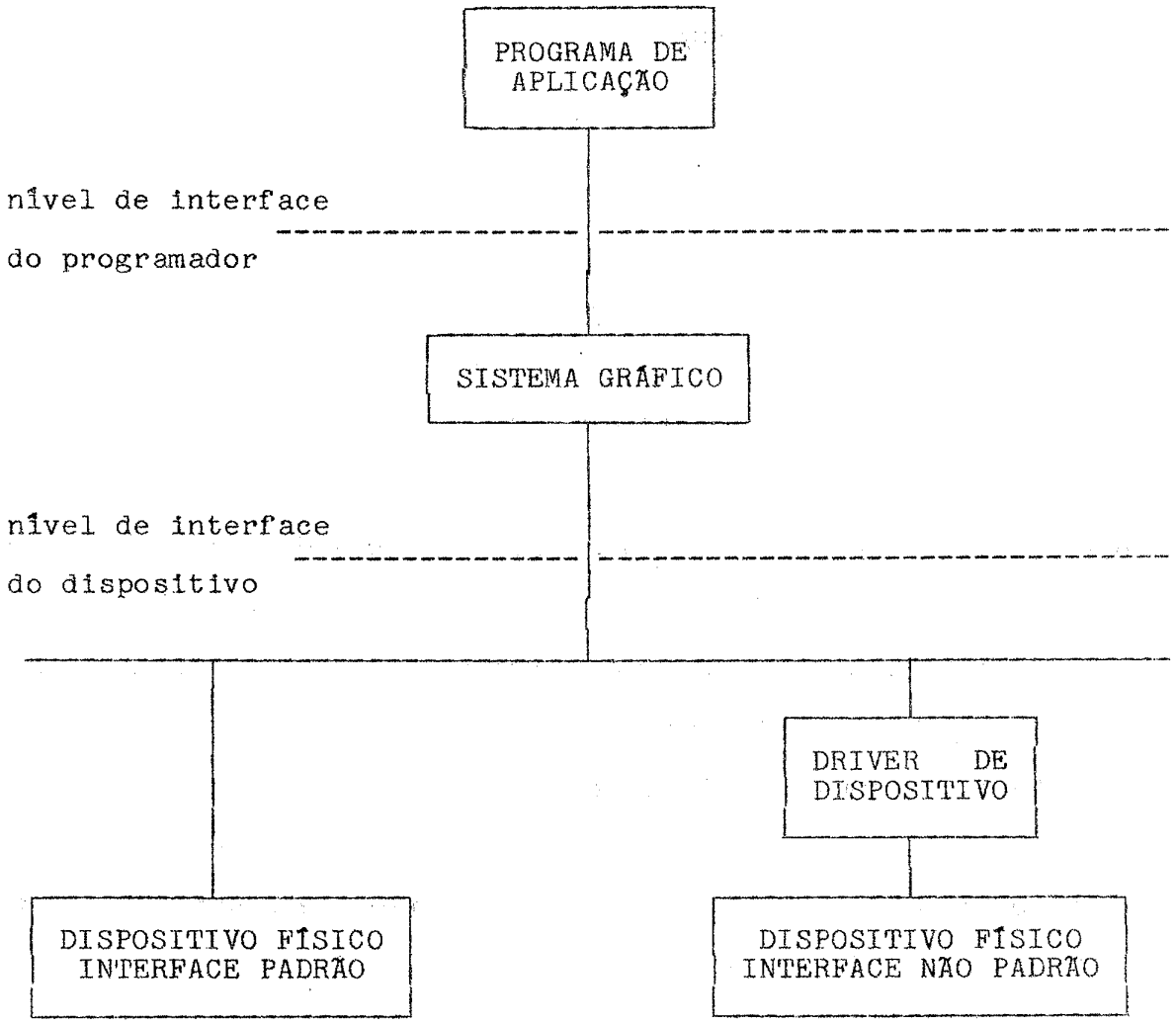
3.6 ESFORÇOS ATUAIS DE PADRONIZAÇÃO

Os esforços atuais de padronização focalizam dois níveis principais de "interface" : a "interface" do programador e a "interface" de dispositivo.

A "interface" do programador se refere ao modelo conceitual e à sintaxe que o programador usa quando incorpora funções grá-

ficas num programa de aplicação. A "interface" de dispositivo se refere ao protocolo usado para a comunicação das funções independentes de dispositivo com as dependentes - às vezes chamada "interface DI/DD". A "interface" do programador padroniza a sequência de chamada e as funções de uma biblioteca de procedimentos gráficos, enquanto a "interface" de dispositivo define um protocolo dispositivo-"driver" que é consistente para todos os dispositivos gráficos.

A figura abaixo ilustra esses conceitos :



3.6.1 Padronização da "Interface" do Programador

Existem duas grandes propostas de padronização neste nível, o CORE e o GKS, descritos nos capítulos subsequentes. O GKS é, possivelmente, a principal proposta; teve origem no DIN, da Alemanha Ocidental, fortemente influenciado pela versão inicial do CORE, de origem americana, e foi desenvolvido com a participação da comunidade de Computação Gráfica internacional, incluindo o próprio ANSI nos EUA. O GKS foi adotado como padrão internacional pela ISO em 1982 e, atualmente, encontra-se em processo de padronização também nos EUA.

3.6.2 Padronização da "Interface" do Dispositivo

Dois novos padrões estão focalizando o nível de interface "hardware"-"driver". Um desses, o "North American Presentation-Level-Protocol Syntax" (NAPLPS), foi desenvolvido por uma equipe na "Bell Laboratories" como uma extensão dos desenvolvimentos gráficos para o sistema de videotexto "Canadian Teledon". NAPLPS - lê-se "naplips" - foi adotado como um padrão para a transmissão de texto e informação gráfica em linhas de telecomunicação. Em algumas aplicações gráficas, o NAPLPS ficará, provavelmente, embutido em outra "interface" de dispositivo, mais geral, chamada "Virtual Device Interface" (VDI).

O padrão VDI está sendo desenvolvido como uma "interface" padrão entre "software" independente de dispositivo e dispositivos gráficos pelo comitê técnico ANSI/X3H3. O VDI faz com que todos os dispositivos pareçam dispositivos gráficos virtuais idênticos através da definição de um protocolo de entrada/saída padrão. As características específicas do dispositivo gráfico são isoladas num módulo de "software" que faz o mapeamento "driver"-dispositivo. Essa técnica foi usada inicialmente por fabricantes individuais de forma a tornar os seus produtos compatíveis com a maior variedade de dispositivos possível, analogamente aos sistemas operacionais UNIX e CP/M para microcomputadores, que "interfaceiam" uma grande variedade de configurações de "hardware". O VDI expande esse conceito propiciando uma compatibilidade potencial em escala industrial.

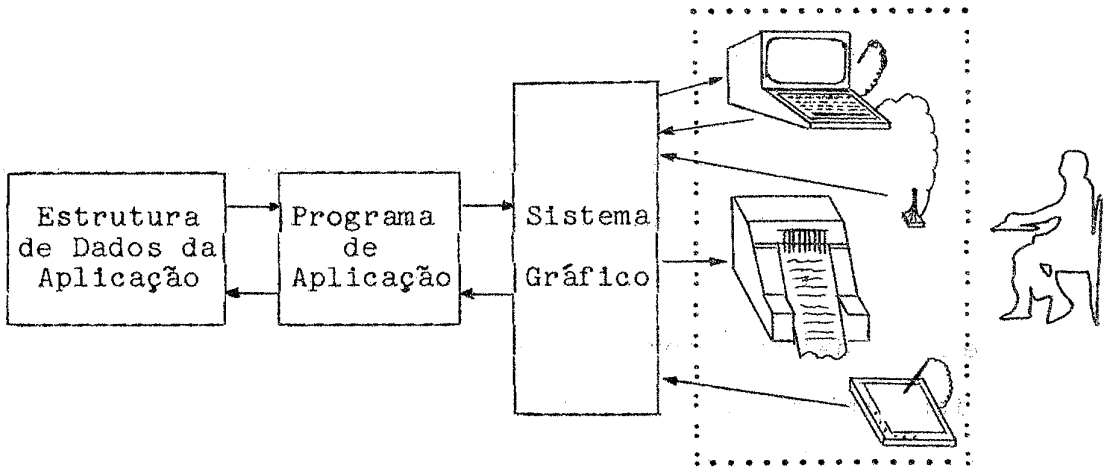
Para o fabricante de equipamento gráfico, a adoção desse padrão significa que o "driver" VDI para um dispositivo específico só precisaria ser escrito uma vez. Todas as aplicações gráficas de acordo com o VDI poderiam se comunicar com o dispositivo via "driver" do dispositivo padrão. Os benefícios se tornariam mais evidentes à medida em que os fabricantes de equipamentos e semiçondutores começassem a implementar mais funções do "driver" em "hardware", movendo de fato a "interface" VDI para o próprio dispositivo.

4. CORE

Logo após a conferência de Seillac, em 1976, foi criado o "Graphics Standards Planning Committee" (GSPC) do "Special Interest Group on Graphics" (SIGGRAPH) da ACM. Em 1977, o GSPC apresentou o primeiro esboço de um padrão para um sistema de subrotinas gráficas na conferência anual do SIGGRAPH, em San Jose, California, o qual ficou conhecido como "sistema CORE". Essa proposta foi refinada e reapresentada em agosto de 1979, no SIGGRAPH'79.

4.1 O MODELO DO PROGRAMADOR

A figura abaixo mostra esquematicamente o modelo do programador que orientou o projeto do sistema CORE :



A componente de "hardware" é um dispositivo de exibição (também chamado de "display", unidade de exibição ou terminal gráfico) que consiste, por sua vez, de uma componente de entrada e uma componente de saída, e é ligado a um computador hospedeiro. As figuras são exibidas na componente de saída chamada tela de exibição ou superfície de visualização e o programa de aplicação recebe informações do operador através da componente de entrada ou componente de interação, que consiste de um conjunto de dispositivos lógicos de entrada.

O "software" tem três componentes. A primeira é o programa de aplicação, que tipicamente armazena/recupera dados na/da segunda componente, a estrutura de dados da aplicação, e envia comandos gráficos para a terceira componente, o sistema gráfico.

Nessa estrutura, o programa de aplicação possibilita ao operador especificar como construir e modificar objetos e indicar que vistas devem ser exibidas na superfície de visualização, através de um diálogo interativo operador-computador via dispositivos de entrada.

O operador especifica ao programa de aplicação que objetos ele deseja visualizar e/ou manipular. Esses objetos podem ser concretos ou abstratos e visualizados num mundo bidimensional ou tridimensional (exemplo: uma função matemática, um circuito, uma molécula). Inicialmente, o programa de aplicação constrói um modelo de aplicação desses objetos, representando as propriedades importantes dos objetos relevantes a uma dada aplicação ou conjunto de aplicações e, na maioria das vezes, armazena esse objeto

na estrutura de dados da aplicação. As descrições que a estrutura de dados armazena contêm, geralmente, coordenadas geométricas que definem a forma das componentes dos objetos, atributos de objetos como cor, estilo de linha, etc., e informação não geométrica (textual ou numérica) relevante. Finalmente, o modelo é descrito ao sistema gráfico pelo programa de aplicação para que ele possa então calcular e exibir a vista desejada do objeto na superfície de visualização.

4.2 A ESTRUTURA DO SISTEMA CORE

É sempre útil ao projeto de qualquer sistema de computação a criação de um modelo simples desse sistema. Dessa forma, é possível fornecer ao programador um conjunto simples de funções, construído sobre um conjunto igualmente simples de idéias abstratas.

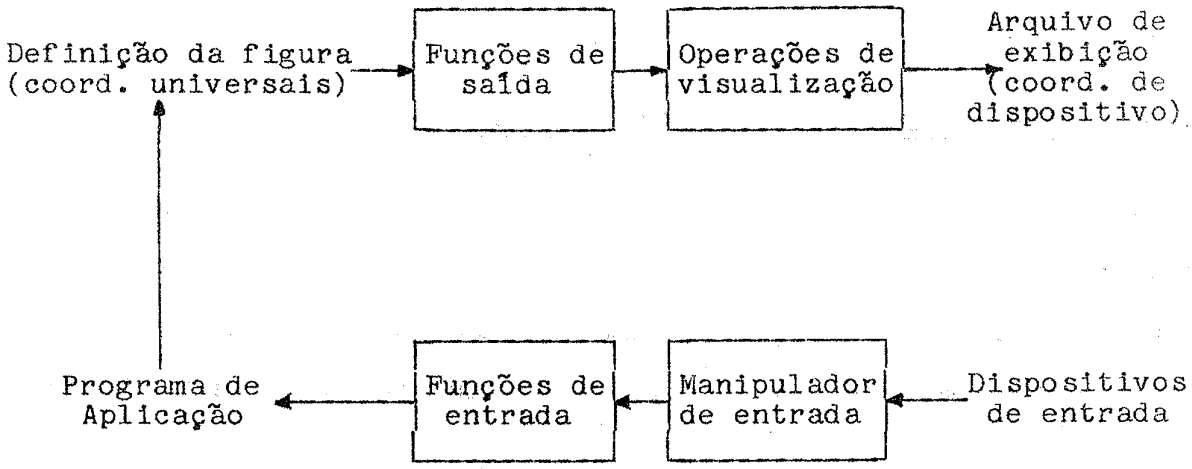
Foram os seguintes os conceitos básicos utilizados para a definição da estrutura do sistema CORE :

- separação das funções de entrada e de saída;
- minimização das diferenças entre as saídas num "plotter" e num dispositivo interativo;
- o conceito de dois sistemas de coordenadas, o sistema de coordenadas universais, no qual o objeto a ser exibido é construído, e o sistema de coordenadas do dispositivo no qual a imagem a ser exibida é representada;
- o conceito de arquivo de exibição ("display file"),

contendo informações em coordenadas do dispositivo, utilizado por qualquer sistema gráfico interativo;

- o conceito de segmentos do arquivo de exibição, mutuamente independentes e podendo ser manipulados como uma unidade;
- o fornecimento de funções para transformar dados de coordenadas universais para coordenadas de dispositivo, através da chamada de operações de visualização.

A figura abaixo mostra a estrutura geral do sistema CORE, construída em torno dos conceitos acima descritos.



O programa de aplicação cria uma definição dos objetos a serem exibidos; essa definição pode ser armazenada numa estrutura de dados específica da aplicação ou pode ser passada diretamente ao sistema CORE. Em ambos os casos, a definição é representada em coordenadas universais no momento em que é passada ao sistema. Em seguida, passa por uma operação de visualização que usa meca-

nismos de "hardware" ou "software" para criar uma representação em coordenadas do dispositivo. Tipicamente, a operação de visualização inclui uma operação de cerceamento ("clipping") que define a porção do espaço de coordenadas universais a ser visualizado. A definição em coordenadas de dispositivo é armazenada num arquivo de exibição com dois objetivos: permitir que a imagem num dispositivo do tipo "refresh" possa ser mantida a partir da definição em coordenadas do dispositivo, e que se possa alterar segmentos individuais de uma imagem sem regenerar o restante. A entrada de dados via "console" do operador fará com que o programa de aplicação calcule novos valores de dados e/ou modifique a figura na tela através de nova chamada de funções de saída.

4.3 INDEPENDÊNCIA LÓGICA DE DISPOSITIVO

O sistema CORE fornece facilidades que protegem o programador de aplicação das características específicas do "hardware". Essa proteção, estratégia fundamental para alcançar a portabilidade, está num nível funcional e não acarreta custo de "software" maior.

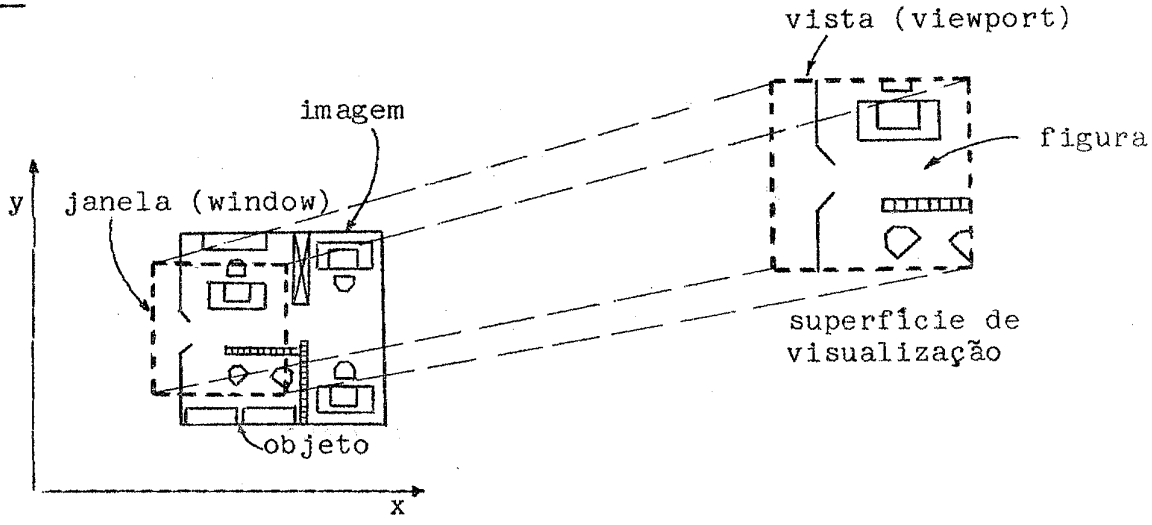
O programador descreve um mundo gráfico para o CORE em coordenadas universais e especifica, além disso, em coordenadas de dispositivo normalizadas, o local onde a vista de um objeto deve ser colocada nas superfícies de visualização lógicas. Analogamente, o programador especifica o uso temporal, pelo operador, de dispositivos de entrada lógicos, sem se preocupar com os protocolos dependentes do "hardware" dos dispositivos reais.

O mapeamento dos dispositivos de entrada lógicos e dispositivos de saída lógicos (superfícies de visualização) em dispositivos físicos para uma configuração específica é manipulado pela implementação do sistema CORE para essa configuração de "hardware". Além disso, alguns dispositivos de entrada físicos são associados a dispositivos de saída físicos específicos (por exemplo, teclado e vídeo), o que não precisa ocorrer necessariamente entre dispositivos de entrada lógicos e superfícies de visualização.

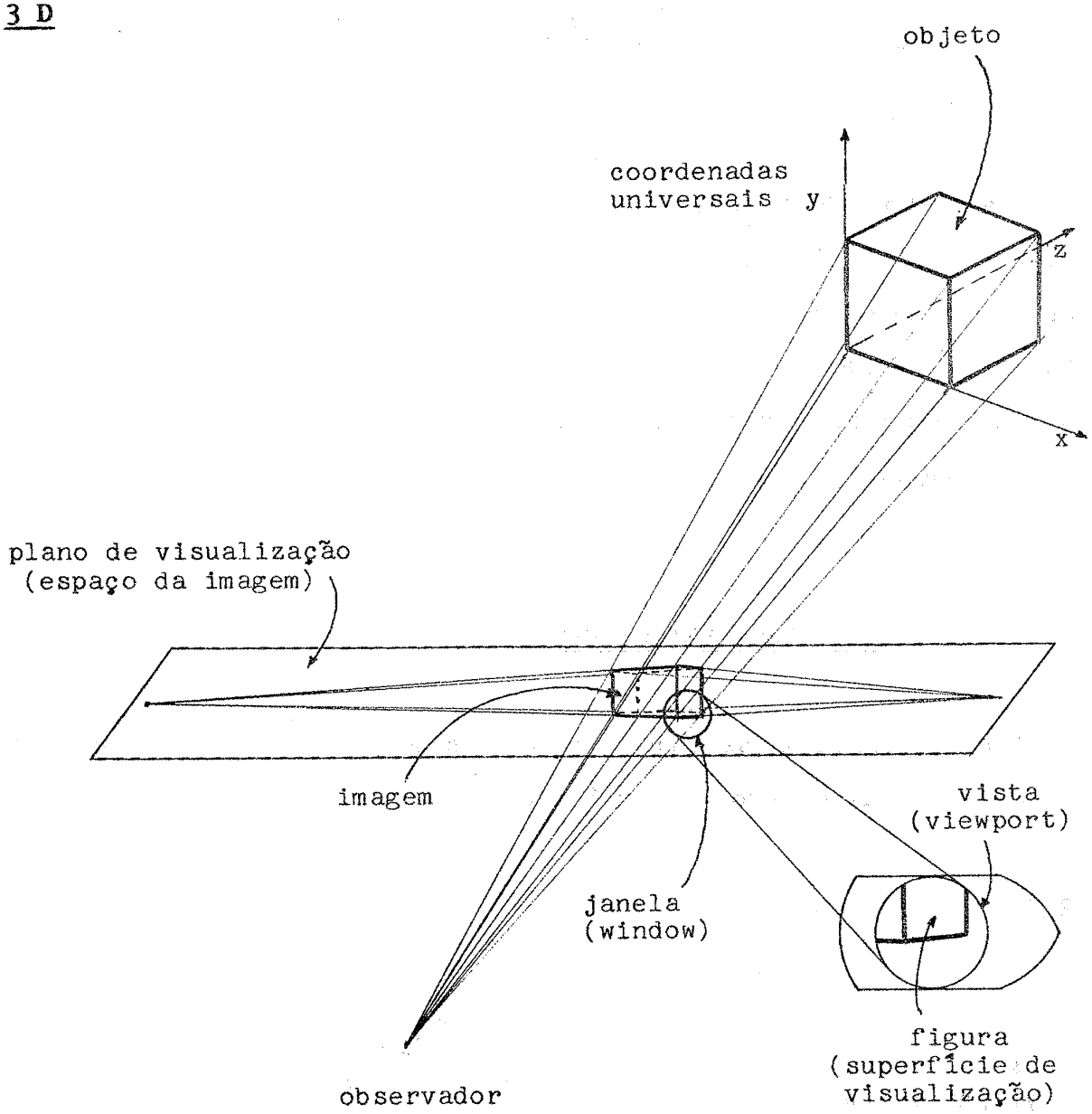
4.4 O MODELO DO PROBLEMA GRÁFICO

As figuras a seguir mostram os modelos bidimensional e tridimensional do problema gráfico.

2 D



3 D



Como se pode observar, o primeiro caso (2D) é um caso particular do segundo (3D).

4.5 A ESPECIFICAÇÃO FUNCIONAL DO SISTEMA CORE

Os conceitos apresentados a seguir referem-se à proposta CORE/79, e são fruto do refinamento dos conceitos da proposta de 1977. A principal diferença entre as duas propostas foi a incorporação de extensões para dispositivos "raster".

4.5.1 Primitivas de Saída

O mundo gráfico que o programador de aplicação descreve para o sistema CORE consiste de um ou mais objetos. A descrição desses objetos é feita em coordenadas universais através de chamadas de funções primitivas de saída bidimensionais ou tridimensionais que descrevem movimentos do cursor, linhas, sequência de linhas, marcas, sequência de marcas e cadeias de caracteres de um texto. Uma chamada de uma dessas funções gera uma primitiva de saída, cuja aparência é afetada por valores de atributos de primitivas como cor, intensidade, estilo de linha, largura de linha, fonte de caracteres, espaçamento entre caracteres, símbolo de marca, entre outros. Cada primitiva pode ter a ela associado um nome, chamado identificador de "pick", para fins de identificação em operação de entrada.

O sistema CORE suporta as seguintes primitivas de saída : LINE, POLYLINE, MARKER, POLYMARKER e TEXT. A LINE é uma linha reta que une dois pontos. Uma POLYLINE é uma sequência de linhas ligadas. Pode-se encarar a primitiva MARKER como um caracter individual que é exibido com tamanho e orientação fixos. Uma

POLYMARKER é um conjunto de marcas com mesmo símbolo. A primitiva de saída TEXT exibe um símbolo para cada caracter especificado numa cadeia de caracteres.

Todas as operações primitivas utilizam coordenadas universais e, com exceção da primitiva TEXT, afetam a posição corrente ("current position" - CP), um valor mantido pelo sistema que define a localização de um cursor no sistema de coordenadas universais. A CP é inicializada na origem do sistema de coordenadas universais e é utilizada para a determinação da posição inicial das primitivas de saída. O sistema também fornece uma função MOVE para alterar o valor da CP sem produzir saída gráfica.

Por conveniência de notação na especificação de coordenadas universais, o sistema permite que os pontos sejam especificados em coordenadas relativas às coordenadas da CP, além da sua especificação em coordenadas absolutas.

A lista abaixo mostra as funções primitivas de saída com seus argumentos.

```
MOVE_ABS_2 (x,y)
MOVE_ABS_3 (x,y,z)
MOVE_REL_2 (dx,dy)
MOVE_REL_3 (dx,dy,dz)

LINE_ABS_2 (x,y)
LINE_ABS_3 (x,y,z)
LINE_REL_2 (dx,dy)
LINE_REL_3 (dx,dy,dz)

POLYLINE_ABS_2 (x_array,y_array,n)
POLYLINE_ABS_3 (x_array,y_array,z_array,n)
POLYLINE_REL_2 (dx_array,dy_array,n)
POLYLINE_REL_3 (dx_array,dy_array,dz_array,n)
```

```
MARKER_ABS_2 (x,y)
MARKER_ABS_3 (x,y,z)
MARKER_REL_2 (dx,dy)
MARKER_REL_3 (dx,dy,dz)

POLYMARKER_ABS_2 (x_array,y_array,n)
POLYMARKER_ABS_3 (x_array,y_array,z_array,n)
POLYMARKER_REL_2 (dx_array,dy_array,n)
POLYMARKER_REL_3 (dx_array,dy_array,dz_array,n)

TEXT (character_string)
```

4.5.2 Segmentação

Todas as primitivas gráficas de saída devem ser colocadas, pelo sistema, num segmento especificado pelo programador de aplicação. Cada segmento define uma imagem, que é uma vista do objeto, e que é parte da figura exibida na superfície de visualização. A descrição de um objeto consiste, então, da criação - abertura - de um segmento, uma sequência de chamadas de funções primitivas de saída e do fechamento do segmento.

No CORE, existem dois tipos de segmentos: retidos e temporários. Os segmentos retidos possuem nome; a distribuição das primitivas de saída associadas a um objeto em diversos segmentos retidos permite que o programador manipule seletivamente partes de uma figura a ele associada, através da exclusão e recriação desses segmentos, alterando suas imagens. Os segmentos temporários são utilizados em aplicações nas quais as imagens só necessitam ser exibidas uma vez; as primitivas colocadas em segmentos temporários não são armazenadas.

Atributos dinâmicos de segmentos retidos afetam as imagens definidas pelos segmentos da mesma forma que os atributos de

primitivas afetam as primitivas de saída. A imagem definida por um segmento retido pode ser controlada quanto à visibilidade, "highlighting", detectabilidade por dispositivos de entrada e transformação da imagem (escala, rotação e translação). Atributos de segmentos podem ser alterados pelo programador depois da criação dos mesmos. Segmentos temporários não possuem atributos.

Além de serem identificadas pelo nome do segmento retido onde foram colocadas, as primitivas de saída também podem ser identificadas por um "pick identifier", que é especificado como um atributo de primitiva. Pode-se atribuir o mesmo identificador a mais de uma primitiva de saída, o que permite selecioná-las como uma unidade.

Abaixo são listadas as funções de segmentação fornecidas pelo sistema CORE.

```
CREATE_RETAINED_SEGMENT (segment_name)
CLOSE_RETAINED_SEGMENT
DELETE_RETAINED_SEGMENT (segment_name)
DELETE_ALL_RETAINED_SEGMENTS
RENAME_RETAINED_SEGMENT (segment_name,new_name)

CREATE_TEMPORARY_SEGMENT
CLOSE_TEMPORARY_SEGMENT
```

4.5.3 Atributos

Atributos especificam características gerais de segmentos retidos e de primitivas de saída; essas características são representadas pelos valores dos atributos. Por exemplo, o atributo LINESTYLE é uma característica das primitivas LINE e

POLYLINE, e tem valores como "sólido", "pontilhado", "tracejado". Analogamente, o atributo VISIBILITY é uma característica de um segmento retido e tem os valores "ligada" ou "desligada".

Atributos podem ser estáticos ou dinâmicos. Atributos estáticos especificam características imutáveis de primitivas de saída ou de segmentos retidos; tais características não se alteram durante a vida dessas entidades. Atributos dinâmicos especificam características que podem ser alteradas durante a vida dessas entidades. Como o segmento é a menor unidade de modificação, as primitivas de saída possuem apenas atributos estáticos. A única forma do programador de aplicação obter o efeito de alterar um certo valor de atributo de uma primitiva é recriar o segmento que contém essa primitiva, usando uma primitiva análoga com o novo valor do atributo. Segmentos retidos possuem os atributos dinâmicos VISIBILITY, HIGHLIGHTING, DETECTABILITY, IMAGE_TRANSLATE e IMAGE_TRANSFORMATION e apenas um atributo estático, IMAGE_TRANSFORMATION_TYPE.

O sistema CORE fornece um conjunto de valores usuais para todos os atributos estáticos e dinâmicos. São atribuídos valores "default" na inicialização do sistema que podem ser explicitamente alterados ou consultados a qualquer tempo até a desativação do sistema. Quando uma primitiva de saída é gerada ou um segmento retido é criado, são associados os valores correntes dos atributos que lhe sejam significativos. Tipicamente, várias primitivas são geradas e vários segmentos retidos são criados entre alterações de valores correntes de atributos. Valores de atributos dinâmicos de segmentos retidos existentes podem ser alterados

explicitamente ou consultados a qualquer tempo, durante a vida desses segmentos.

A seguir são listadas as funções que permitem estabelecer valores para os diversos atributos.

Atributos de linha

```
SET_COLOR (color)
SET_INTENSITY (intensity)
SET_LINESTYLE (linestyle)
SET_LINEWIDTH (linewidth)
SET_PEN (pen) ... agrupa os quatro anteriores
```

Atributos de texto

```
SET_FONT (font)
SET_CHARSIZE (charwidth,charheight)
SET_CHARPLANE (dx_plane,dy_plane,dz_plane)
SET_CHARUP_2 (dx_charup,dy_charup)
SET_CHARUP_3 (dx_charup,dy_charup,dz_charup)
SET_CHARPATH (charpath)
SET_CHARSPACE (charspace)
SET_CHARJUST (charjust)
SET_CHARPRECISION (charprecision)
```

Atributo de marca

```
SET_MARKER_SYMBOL (symbol)
```

Atributo de identificação da primitiva

```
PICK_ID (id)
```

Conjuntos dos atributos de primitivas

```
SET_PRIMITIVE_ATTRIBUTES_2 (primitive_attribute_array_2)
SET_PRIMITIVE_ATTRIBUTES_3 (primitive_attribute_array_3)
```

Atributo estático de segmento retido

```
SET_IMAGE_TRANSFORMATION_TYPE (type)
```

Atributos dinâmicos para todos os segmentos retidos

```
SET_VISIBILITY (visibility)
SET_HIGHLIGHTING (highlighting)
SET_DETECTABILITY (detectability)
SET_IMAGE_TRANSLATE_2 (tx,ty)
SET_IMAGE_TRANSFORMATION_2 (sx,sy,a,tx,ty)
SET_IMAGE_TRANSFORMATION_3 (sx,sy,sz,ax,ay,az,tx,ty,tz)
```

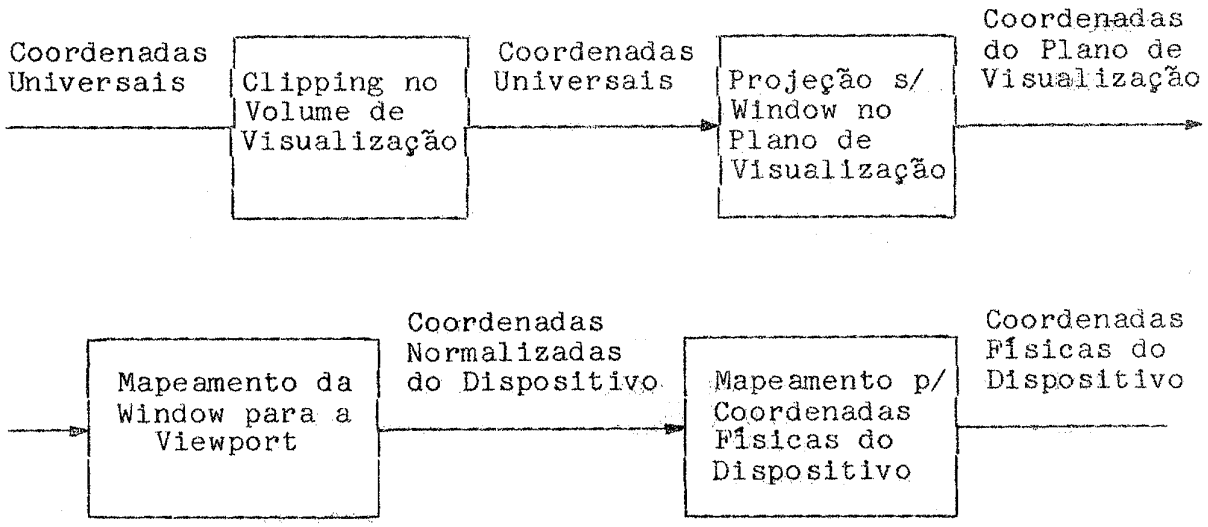
Atributos dinâmicos para um segmento retido

```
SET_SEG_VISIBILITY (seg_name,visibility)
SET_SEG_HIGHLIGHTING (seg_name,highlighting)
SET_SEG_DETECTABILITY (seg_name,detectability)
SET_SEG_IMAGE_TRANSLATE_2 (seg_name,tx,ty)
SET_SEG_IMAGE_TRANSF_2 (seg_name,sx,sy,a,tx,ty)
SET_SEG_IMAGE_TRANSF_3 (seg_name,sx,sy,sz,ax,ay,az,tx,ty,tz)
```

4.5.4 Operações de Visualização e Transformações de Coordenadas

O modelo utilizado para a representação de objetos na superfície de visualização desejada é o da câmera sintética. O programador de aplicação fornece a descrição de uma cena do mundo real consistindo de um ou mais objetos num mundo imaginário ou sintético, bidimensional ou tridimensional. A câmera sintética produz então uma vista do(s) objeto(s) nesse mundo, que depende, naturalmente, de como a câmera foi instalada e qual a sua posição em relação aos objetos. Como numa máquina Polaroid, o instantâneo é revelado e torna-se disponível quase imediatamente para a exibição na superfície de visualização.

O processo geral de criação de uma imagem numa superfície de visualização pode ser esquematizado como mostra a figura a seguir.



Em duas dimensões, a operação de visualização é completamente especificada por uma janela ("window") em coordenadas universais e uma vista ("viewport") na superfície de visualização. A janela pode ser qualquer retângulo no plano de coordenadas universais XY; são permitidas janelas inclinadas em relação aos eixos de coordenadas. A vista é um retângulo especificado em coordenadas normalizadas do dispositivo, e seus lados são paralelos aos limites horizontais e verticais da superfície de visualização. A janela é utilizada para o cerceamento ("clipping") do objeto e para determinar o mapeamento janela-vista, ou seja, a porção do objeto limitada pela janela é mapeada para a porção do espaço de coordenadas normalizadas do dispositivo limitada pela vista.

A operação de visualização para objetos tridimensionais é mais complexa, já que inclui uma projeção para um plano de visualização, além do cerceamento e do mapeamento para a superfície de visualização. Ela corresponde à especificação da posição da

câmera sintética, seu tipo de projeção (paralela ou perspectiva) e o local em que a vista do objeto (a imagem) deve ser mostrada na superfície de visualização; assim, para se obter uma nova vista, deve-se mover a câmera para uma nova posição em relação ao objeto.

No sistema CORE uma projeção é especificada por um plano de projeção dentro do mundo gráfico, às vezes chamado de plano de visualização, e um centro de projeção no caso de projeção perspectiva ou uma direção de projeção no caso de projeção paralela.

Dois tipos de cerceamento fazem parte da visualização tridimensional. O primeiro, o cerceamento de profundidade, limita uma região entre dois planos paralelos ao plano de projeção. O segundo cerceamento é feito através da especificação de uma janela no plano de projeção. Essas duas etapas de cerceamento definem uma região visível, chamada volume de visualização em 3D. Para a projeção perspectiva, o volume de visualização é uma pirâmide retangular truncada. Para a projeção paralela, é um paralelepípedo finito.

O mapeamento da janela para a vista na superfície de visualização é análogo ao mapeamento no caso bidimensional.

A visualização bidimensional é um subconjunto da visualização tridimensional. As capacidades funcionais para visualização são tais que o programador de uma aplicação bidimensional não necessita conhecer as construções da visualização tridimensional.

Após uma operação de visualização, a imagem resultante pode ser mudada de escala, rodada ou transladada através da transformação de imagem. Em termos do modelo da câmera sintética, essa facilidade corresponde a aumentar ou reduzir, rodar e posicionar o instantâneo antes de colocá-lo na superfície de visualização.

São mostradas, a seguir, as funções fornecidas pelo sistema CORE para as operações de visualização e transformações de coordenadas.

Funções bidimensionais

```
SET_WINDOW (xmin,xmax,ymin,ymax)
SET_VIEW_UP_2 (dx_up,dy_up)
SET_NDC_SPACE_2 (width,height)
SET_VIEWPORT_2 (xmin,xmax,ymin,ymax)
MAP_NDC_TO_WORLD_2 (ndc_x,ndc_y,x,y)
MAP_WORLD_TO_NDC_2 (x,y,ndc_x,ndc_y)
```

Funções tridimensionais

```
SET_VIEW_REFERENCE_POINT (x_ref,y_ref,z_ref)
SET_VIEW_PLANE_NORMAL (dx_norm,dy_norm,dz_norm)
SET_VIEW_PLANE_DISTANCE (view_distance)
SET_VIEW_DEPTH (front_distance,back_distance)
SET_PROJECTION (projection_type,dx_proj,dy_proj,dz_proj)
SET_WINDOW (umin,umax,vmin,vmax)
SET_VIEW_UP_3 (dx_up,dy_up,dz_up)
SET_NDC_SPACE_3 (width,height,depth)
SET_VIEWPORT_3 (xmin,xmax,ymin,ymax,zmin,zmax)
SET_VIEWING_PARAMETERS (viewing_parameters_array)
MAP_NDC_TO_WORLD_3 (ndc_x,ndc_y,ndc_z,x,y,z)
MAP_WORLD_TO_NDC_3 (x,y,z,ndc_x,ndc_y,ndc_z)
```

Outras funções

```
SET_WINDOW_CLIPPING (on_off)
SET_FRONT_PLANE_CLIPPING (on_off)
SET_BACK_PLANE_CLIPPING (on_off)
SET_COORDINATE_SYSTEM_TYPE (type)
SET_WORLD_COORDINATE_MATRIX_2 (matrix_2)
SET_WORLD_COORDINATE_MATRIX_3 (matrix_3)
```

4.5.5 Primitivas de Entrada

Os dispositivos lógicos de entrada no sistema CORE correspondem a abstrações das propriedades dos dispositivos físicos existentes. O "hardware" de entrada que realmente suporta esses dispositivos lógicos é dependente de implementação; as propriedades funcionais é que são importantes para a portabilidade de programas e podem ser suportadas via simulação por "software", se necessário.

O sistema CORE suporta seis classes de dispositivos lógicos para entrada pelo operador, descritas a seguir, com os respectivos exemplos físicos típicos.

- PICK : identifica uma primitiva de saída através do nome do segmento e do "pick identifier". Exemplo : "light pen".

- LOCATOR : fornece uma posição na superfície de visualização, em coordenadas normalizadas do dispositivo. Exemplos : "tablet" e "joystick".

- STROKE : fornece uma série de posições na superfície de visualização, em coordenadas normalizadas do dispositivo. Exemplo : estilete de "tablet".

- VALUATOR : fornece um valor escalar. Exemplo : "dial" de controle (potenciômetro).

- KEYBOARD : permite a entrada de um texto alfanumérico. Exemplo : teclado.

- BUTTON : permite a escolha lógica de alternativas.
  Exemplo : teclas programáveis.

Todos os dispositivos devem ser inicializados e habilitados para o uso. Cada classe de dispositivo de entrada tem uma variedade de "feedbacks" (ecos) definidos pelo sistema, que podem ser solicitados pelo programador de aplicação.

Os dispositivos de entrada lógicos podem causar evento ou serem amostrados. Dispositivos do tipo PICK, KEYBOARD, BUTTON e STROKE causam um evento quando, habilitados, são utilizados pelo operador; a cada entrada é colocado um relatório do evento numa fila de eventos, que pode ser consultada pelo programador de aplicação. Dispositivos causadores de evento não podem ser amostrados. Reciprocamente, o uso de dispositivos do tipo LOCATOR e VALUATOR não causam um evento; quando habilitados, esses dispositivos podem ser amostrados pelo programa de aplicação. Além disso, o programador de aplicação pode associar dispositivos causadores de evento com amostrados de forma que, quando ocorrer um evento, os valores dos dispositivos amostrados associados, habilitados, possam ser colocados no relatório do evento.

Também existe disponível um outro estilo de programação de entrada, a entrada síncrona. Quando um programa de aplicação precisa de um valor do operador, o programa de aplicação chama uma função que espera, por algum tempo, o valor de entrada desejado para retornar. Com a entrada síncrona, o programador de aplicação deve inicializar os dispositivos de entrada antes de utilizá-los, mas não precisa habilitá-los, criar associação ou

manipular filas de eventos.

A seguir são listadas as funções que permitem a manipulação de entrada no sistema CORE.

Funções de inicialização e habilitação

```
INITIALIZE_DEVICE (device_class,device_num)
INITIALIZE_GROUP (device_class,device_num_array,n)
ENABLE_DEVICE (device_class,device_num)
ENABLE_GROUP (device_class,device_num_array,n)
DISABLE_DEVICE (device_class,device_num)
DISABLE_GROUP (device_class,device_num_array,n)
DISABLE_ALL
TERMINATE_DEVICE (device_class,device_num)
TERMINATE_GROUP (device_class,device_num_array,n)
```

Dispositivos amostrados

```
READ_LOCATOR_2 (locator_num,x,y)
READ_LOCATOR_3 (locator_num,x,y,z)
READ_VALUATOR (valuator_num,value)
```

Manipulação de eventos

```
AWAIT_EVENT (time,event_class,event_num)
FLUSH_DEVICE_EVENTS (event_class,event_num)
FLUSH_GROUP_EVENTS (event_class,event_num_array,n)
FLUSH_ALL_EVENTS
```

Associação de eventos

```
ASSOCIATE (event_class,event_num,sampl_class,sampl_num)
DISASSOCIATE (event_class,event_num,sampl_class,sampl_num)
DISASSOCIATE_DEVICE (device_class,device_num)
DISASSOCIATE_GROUP (device_class,device_num_array,n)
DISASSOCIATE_ALL
```

Acesso a dados do relatório de eventos

```
GET_PICK_DATA (segment_name,pick_id)
GET_KEYBOARD_DATA (input_string,num_input)
GET_STROKE_DATA_2 (array_size,x_array,y_array,n)
GET_STROKE_DATA_3 (array_size,x_array,y_array,z_array,n)
GET_LOCATOR_DATA_2 (locator_num,x,y)
GET_LOCATOR_DATA_3 (locator_num,x,y,z)
GET_VALUATOR_DATA (valuator_num,value)
```

Funções de entrada síncrona

```
AWAIT_ANY_BUTTON (time,button_num)
AWAIT_PICK (time,pick_num,segment_name,pick_id)
AWAIT_KEYBOARD (time,keyboard_num,input_string,length)
AWAIT_STROKE_2 (time,str_num,arr_size,x_arr,y_arr,n)
AWAIT_STROKE_3 (time,str_num,arr_size,x_arr,y_arr,z_arr,n)
AWAIT_ANY_BUTTON_GET_LOCATOR_2 (time,loc_num,but_num,x,y)
AWAIT_ANY_BUTTON_GET_LOCATOR_3 (time,loc_num,but_num,x,y,z)
AWAIT_ANY_BUTTON_GET_VALUATOR (time,val_num,but_num,value)
```

Funções de eco

```
SET_ECHO (device_class,device_num,echo_type)
SET_ECHO_SEGMENT (device_class,device_num,segment_num)
SET_ECHO_SURFACE (device_class,device_num,surface_name)
SET_ECHO_POSITION (device_class,device_num,echo_x,echo_y)
```

Estabelecimento de características dos dispositivos

```
SET_PICK (num,aperture)
SET_KEYBOARD (num,buffer_size,initial_string,cursor_start)
SET_BUTTON (num,prompt_switch)
SET_ALL_BUTTONS (prompt_switch)
SET_STROKE (num,buffer_size,distance,time)
SET_LOCATOR_2 (num,loc_x,loc_y)
SET_LOCATOR_3 (num,loc_x,loc_y,loc_z)
SET_LOCPORT_2 (locator_num,xmin,xmax,ymin,ymax)
SET_LOCPORT_3 (locator_num,xmin,xmax,ymin,ymax,zmin,zmax)
SET_VALUATOR (num,initial_value,low_value,high_value)
```

4.5.6 Funções de Controle e de Consulta

O funcionamento geral do sistema pode ser controlado de várias formas. São fornecidas funções para inicializar e terminar o sistema, selecionar e controlar superfícies de visualização, controlar mudanças de figuras, controlar quadros e manipular relatórios de erros. A expressão "new-frame action" refere-se à operação necessária para eliminar ou alterar parte de uma figura. O momento da exibição da figura pode ser controlado pelo programa

de aplicação. É fornecida também a capacidade de grupar alterações da figura em lotes ("batches"), o que é útil, por exemplo, para evitar que se apague um dispositivo tipo "storage tube" a cada pequena alteração da figura.

A seguir estão listadas as funções de controle do sistema CORE.

```
INITIALIZE_CORE (outlevel,inlevel,dimension)
TERMINATE_CORE
INITIALIZE_VIEW_SURFACE (surface_name)
TERMINATE_VIEW_SURFACE (surface_name)
SELECT_VIEW_SURFACE (surface_name)
DESELECT_VIEW_SURFACE (surface_name)

SET_IMMEDIATE_VISIBILITY (immediacy)
MAKE_PICTURE_CURRENT
BEGIN_BATCH_OF_UPDATES
END_BATCH_OF_UPDATES
SET_VISIBILITIES (segment_name_array,visibility_array,n)
NEW_FRAME

REPORT_MOST_RECENT_ERROR (error_report)
LOG_ERROR (error_report)
```

Além dessas funções de controle, o sistema fornece uma vasta gama de funções de "INQUIRY", com o objetivo de permitir o máximo de informação ao programa de aplicação. Todas as características estabelecidas pelas funções primitivas de saída, de segmentação, de atributos, de visualização e transformação de coordenadas, de entrada e de controle podem ser consultadas através dessas funções.

4.5.7 Mecanismo de Escape

Embora a proposta do sistema CORE seja fornecer um sistema

gráfico padrão, não há dúvida quanto à necessidade de certas capacidades funcionais não suportadas diretamente pelo sistema, para a implementação eficiente de algumas aplicações. O CORE fornece uma estrutura padrão para essa "interface", de forma a preservar um grau de portabilidade de aplicação não padrão.

Esse mecanismo especial de "interface" é chamado de escape, e fornece meios para que funções não padronizadas sejam acessadas de uma forma padronizada. Tais funções são dependentes da instalação e dos dispositivos. Um exemplo de tal função não padrão é o gerador de curvas suportado por alguns terminais gráficos.

O sistema fornece uma função ESCAPE, presente em todas as implementações CORE mesmo que não suportem funções não padronizadas. Esta é uma forma de se assegurar o processamento correto de erros, quando são transportados programas que utilizam o mecanismo de escape.

Para aumentar a portabilidade de programas que usam essa interface, as chamadas da função ESCAPE devem ser restritas ao mínimo no programa de aplicação. Nesses poucos locais, a chamada específica dessa facilidade deve ser isolada e amplamente comentada, incluindo uma descrição detalhada da função que está sendo utilizada.

A função de escape do sistema CORE é da forma :

ESCAPE (function_name,parameter_count,parameter_list)

4.6 NÍVEIS DE IMPLEMENTAÇÃO DO SISTEMA

O sistema CORE foi projetado para ser utilizado por uma grande gama de aplicações, desde aquelas que necessitam apenas de plotagem estática até às que utilizam interação em tempo real e movimento dinâmico. Além disso, muito dispositivos não possuem algumas facilidades, o que requer um esforço de implementação considerável para simulá-las em "software".

Assim, o projeto de um sistema monolítico, embora desejável para garantir a portabilidade de programas, não é aceitável. Ao invés disso, são especificadas três classes de níveis de implementação, compatíveis e abrangentes, onde os níveis mais altos incluem os mais baixos. O conceito de níveis evita que os implementadores do sistema escolham arbitrariamente as facilidades desejadas, o que ocasionaria a geração de um grande número de hábitos não padrões, ou seja, a destruição do objetivo mais importante : a portabilidade.

As classes de níveis especificadas são: saída, entrada e dimensão do espaço de coordenadas universais.

4.6.1 Níveis de Saída

São três os níveis de saída : saída básica, saída "bufferizada" e saída dinâmica. O nível de saída básica serve para aplicações que não necessitam de modificação seletiva de figuras; o de saída "bufferizada" permite que segmentos selecionados sejam retidos e o de saída dinâmica fornece o conjunto

completo de capacidades funcionais para apresentação de figuras, incluindo transformação de imagem.

A tabela abaixo resume as capacidades funcionais dos três níveis de saída :

Capacidades Funcionais	Nível 1 Básico	Nível 2 Bufferizado	Nível 3 Dinâmico
Primitivas de saída e Atributos de primitivas	sim	sim	sim
Visualização	sim	sim	sim
Controle	sim	sim	sim
Segmentos temporários	sim	sim	sim
Segmentos retidos	não	sim	sim
Atributo "highlighting" de segmento	não	sim	sim
Atributo visibilidade de segmento	não	sim	sim
Atributo transformação de imagem de segmento	não	não	sim
Atributo detectabilidade de segmento	não	sim*	sim*

* - se o nível de entrada for 2 ou 3

4.6.2 Níveis de Entrada

O CORE tem três níveis de entrada : nenhuma entrada, entrada síncrona e entrada completa. O primeiro nível não suporta primitivas de entrada, o nível de entrada síncrona serve para suporte de aplicações que não necessitam de sequências de interação

assíncrona, e o último nível fornece suporte para todas as ferramentas de interação definidas pelo sistema.

A tabela abaixo resume as capacidades funcionais dos níveis de entrada :

Capacidades Funcionais	Nível 1 Nenhuma	Nível 2 Síncrona	Nível 3 Completa
Inicialização de Dispositivo	não	sim	sim
Funções síncronas	não	sim	sim
Controle de eco	não	sim	sim
Habilitação/Desabilitação explícita	não	não	sim
Gerência de fila de eventos	não	não	sim
Funções de leitura de dispositivos amostrados	não	não	sim
Associações	não	não	sim

4.6.3 Níveis de Dimensão

Os níveis de dimensão são apenas dois: bidimensional e tridimensional. O nível 2D oferece apenas primitivas, atributos, operações de visualização e entrada em duas dimensões e o nível 3D fornece todas as capacidades do sistema apropriadas para os níveis de entrada e saída implementados.

4.7 EXTENSÕES PARA A TECNOLOGIA "RASTER"

A proposta CORE de 1977 foi projetada para atender principalmente a dispositivos de desenho de linhas. Foram excluídas várias áreas, entre elas os dispositivos "raster". Após a publicação daquele documento original, foram criados subcomitês do GSPC, visando aprimorar e atualizar a proposta. A versão de 1979 incorporou os resultados do trabalho desses grupos.

O "CORE Extensions Subcommittee", através do seu "Raster and Color Group" tinha como objetivo principal desenvolver extensões para possibilitar utilização de dispositivos "raster", que emergiram rapidamente na época, devido aos grandes progressos na área de circuitos integrados e a grande demanda de gráficos coloridos e sombreados.

Uma das principais características que diferenciam dispositivos tipo "raster" de outros é a sua capacidade de preencher áreas com cor e intensidade controladas. Consequentemente, além de poder desenhar linhas, marcas e textos, o sistema precisa ser capaz de definir e preencher áreas. Essa necessidade foi suprida no sistema CORE estendido, através da capacidade para definir e preencher polígonos planos.

Foram definidas, além disso, funções para especificação de cor e intensidade, já que áreas poligonais podem ser preenchidas com várias técnicas, em cor ou intensidade, dependendo da aplicação e do dispositivo.

Para possibilitar o preenchimento de polígonos com padrões

fixos, foi definido o conceito de "pixel array" e especificadas funções para a sua manipulação. Um padrão é o resultado do mapeamento de um "pixel array" numa superfície de visualização.

Finalmente, foram incluídas funções para a solução do problema de remoção de superfícies escondidas ("hidden surfaces"), natural quando se inclui a definição de superfícies em três dimensões.

São listadas a seguir as funções incluídas na extensão do CORE para dispositivos "raster".

Primitivas de saída

```
POLYGON_ABS_2 (x_array,y_array,n)
POLYGON_ABS_3 (x_array,y_array,z_array,n)
POLYGON_REL_2 (x_array,y_array,n)
POLYGON_REL_3 (x_array,y_array,z_array,n)
```

Atributos

```
SET_POLYGON_INTERIOR_STYLE (interior_style)
SET_POLYGON_EDGE_STYLE (edge_style)
SET_LINE_INDEX (line_index)
SET_FILL_INDEX (fill_index)
SET_TEXT_INDEX (text_index)
SET_VERTEX_INDICES (vertex_indices)
SET_COLOR_MODEL (model)
```

Controles auxiliares

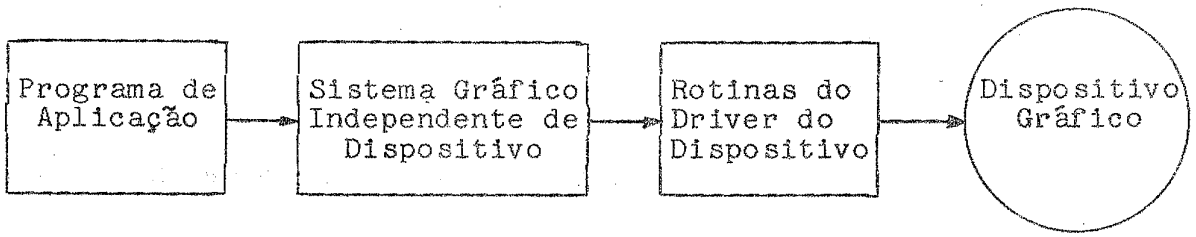
```
SET_BACKGROUND_INDEX (index)
SET_DISPLAY_MODE (mode)
```

Funções de "pixel array"

```
SET_PIXEL_ARRAY (index_array,columns,rows)
SET_PIXEL_PATTERN_ORIGIN_2 (x_abs,y_abs)
```

4.8 A PROPOSTA DO "METAFILE"

A geração de figuras num dispositivo gráfico ocorre geralmente como descreve a figura abaixo:

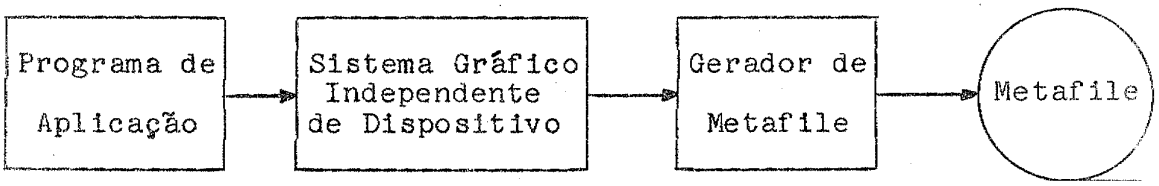


Um outro subcomitê do GSPC, o "Partitioning and Protocols Subcommittee" foi responsável pela definição de um arquivo de exibição ("display file") intermediário, independente de dispositivo, que foi incorporada na proposta de 1979. Esse arquivo, chamado "METAFILE" ou "pseudo display file", permite, entre outras coisas, a transferência de figuras entre duas instalações. A proposta inclui a especificação de formato dos registros e de comandos.

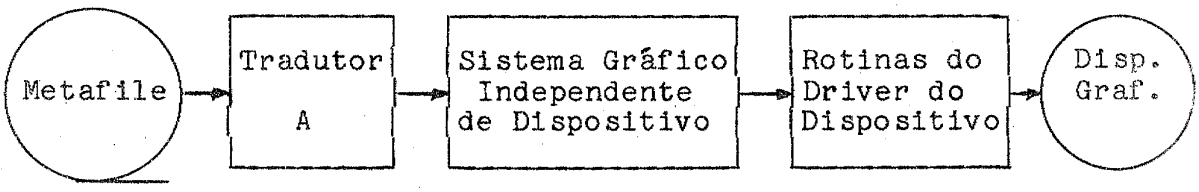
Para a implementação desse conceito, a geração de figuras é executada em duas fases. Foram incluídos um gerador de "metafile", que pode ser parte do sistema gráfico independente de dispositivo, e um tradutor de "metafile" para interpretar os comandos de "metafile" independentes de dispositivo. Esse tradutor pode chamar rotinas do sistema gráfico ou chamar diretamente rotinas do "driver" do dispositivo.

A implementação do conceito de "metafile" pode ser descrito como nas figuras seguintes.

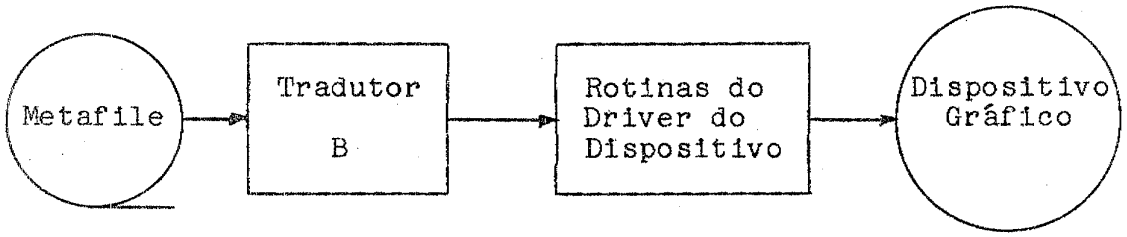
Fase 1 : Geração do "Metafile"



Fase 2 : Tradução do "Metafile"



ou



4.9 GLOSSÁRIO

Este glossário é o apresentado na proposta de 1979. Foi mantida a ordem de apresentação original, e as traduções utilizadas no texto estão entre parênteses. Quando a tradução não foi possível, manteve-se o termo em inglês.

- Operator or Console Operator (Operador ou Operador de Console)
  Usuário de um programa de aplicação interativo que é implementado utilizando o sistema CORE. O operador interage com o programa de aplicação através de uma console de exibição ("display console"), isto é, uma tela de exibição física e uma coleção de dispositivos de entrada.

- Programmer (Programador)
  Usuário do sistema CORE. Pessoa que escreve programas de aplicação gráficos numa linguagem hospedeira, como FORTRAN.

- Device Driver ("Driver" de Dispositivo)
  A parte dependente de dispositivo de uma implementação do sistema CORE que suporta um dispositivo gráfico físico. O "driver" do dispositivo gera saída e manipula interação dependente de dispositivo.

- Modelling System (Sistema de Modelagem)
  Sistema de alto nível para definição de objetos. Um sistema de modelagem descreve objetos para o sistema CORE utilizando coordenadas universais.

- World Coordinate System (Sistema de Coordenadas Universais)
  Sistema de coordenadas cartesianas 3D independente de dispositivo, no qual objetos em 2D ou 3D são descritos para o sistema CORE.

- Normalized Device Coordinate System (Sistema de Coordenadas Normalizadas de Dispositivo)
  Sistema de coordenadas cartesianas 2D ou 3D independente de dispositivo, cujas coordenadas estão no intervalo 0 a 1. Coordenadas normalizadas de dispositivo são usadas na definição de vistas de objetos. Em particular, são usadas para especificação de "viewports", transformação de imagens e entrada obtida de dispositivos "stroke" e "locator".

- Normalized Device Coordinate Space (NDC-space) (Espaço de Coordenadas Normalizadas de Dispositivo)
  Região finita no sistema de coordenadas normalizadas de dispositivo. Define a região máxima utilizável por um programa de aplicação.

- Device Coordinate System or Screen Coordinate System (Sistema de Coordenadas de Dispositivo ou Sistema de Coordenadas de Tela)
Sistema de coordenadas dependente de dispositivo cujas coordenadas são dadas, tipicamente, ou em unidades "raster" inteiras, ou frações entre 0 e 1. "Drivers" de dispositivo mapeiam coordenadas normalizadas de dispositivo em coordenadas de tela.

- Viewing Operation (Operação de Visualização)
Operação que mapeia posições em coordenadas universais para posições em coordenadas normalizadas de dispositivo. Além disso, especifica a porção do espaço de coordenadas universais que deve ser visível.

- Current Position (CP) (Posição Corrente)
Valor do sistema CORE que define a localização do desenho corrente em coordenadas universais. É "setada" na origem do sistema de coordenadas universais na inicialização do sistema. O valor da CP é alterado por chamadas de funções que criam primitivas de saída.

- Object (Objeto)
A unidade gráfica conceitual no programa de aplicação. Vistas de objetos são especificadas por uma operação de visualização e exibidas pelo sistema CORE. Objetos são descritos para o sistema CORE em coordenadas universais em termos de chamadas de funções primitivas de saída e estabelecimento de atributos de primitivas.

- Image (Image)
Uma vista específica de um ou mais objetos ou partes de objetos. É uma parte da figura numa superfície de visualização. Uma operação de visualização age na descrição de um objeto para produzir primitivas de saída num segmento. Primitivas de saída de um segmento, seus valores de atributos e os valores dos atributos do segmento reunidos definem uma imagem.

- View Surface (Superfície de Visualização)
Uma superfície de saída lógica bidimensional. Imagens numa superfície de visualização são desenhadas na superfície de saída física correspondente (por exemplo, uma superfície de "plotter" ou tela de exibição) numa forma dependente de dispositivo pelo "driver" do dispositivo. Pode-se implementar uma pseudo-superfície de visualização que armazena figuras num "metafile".

- Primitive or Output Primitive (Primitiva ou Primitiva de Saída)
Um elemento de figura (por exemplo, uma linha ou um "string" de texto) que tem uma aparência específica. Valores de atributos de primitiva determinam certos aspectos de sua aparência.

- Primitive Attribute (Atributo de Primitiva)
Uma característica geral de uma primitiva de saída. Os atributos de primitiva fornecidos pelo sistema CORE são cor, intensi-

dade, estilo de linha, largura de linha, "pen", "font", tamanho de caracter, plano de caracter, "character up", caminho de caracter, espaço entre caracteres, alinhamento de "string" de caracteres, precisão de caracter, símbolo de marca e identificador de "pick".

- Segment (Segmento)
Coleção ordenada de primitivas de saída que define uma imagem que é parte de uma figura numa superfície de visualização.

- Temporary Segment (Segmento Temporário)
Um segmento sem nome que não tem atributos de segmento. A imagem definida por um segmento temporário permanece visível apenas enquanto informações são adicionadas à figura exibida. A imagem de um segmento temporário desaparece assim que ocorre uma ação de "new frame", ou seja, assim que a informação é removida da figura exibida.

- Retained Segment (Segmento Retido)
Um segmento com nome que tem associados atributos dinâmicos de segmentos retidos que podem ser modificados, alterando consequentemente a imagem do segmento. Para alterar as primitivas de saída em um segmento retido, este deve ser excluído e recriado.

- Segment Attribute (Atributo de Segmento)
Uma característica geral de um segmento retido. O sistema CORE fornece um atributo estático de segmento retido e quatro dinâmicos. O atributo estático especifica que transformações de imagem podem ser aplicadas a um segmento. Os atributos dinâmicos são visibilidade, "highlighting", detectabilidade e transformação de imagem. Os valores dos atributos dinâmicos de um segmento retido podem ser alterados, modificando consequententemente as características desse segmento.

- Image Transformation (Transformação de Imagem)
Um atributo dinâmico de segmento retido que permite que a imagem definida por um segmento apareça com variados tamanhos, orientações e/ou posições na superfície de visualização.

- New Frame Action (Ação "New Frame")
A eliminação de toda a informação temporária e o re-desenho, se necessário, de toda a informação retida visível. Esta ação é implícita em várias funções, como por exemplo tornar um segmento invisível. Em dispositivos para "hardcopy", por exemplo, o meio de gravação é avançado para uma nova área de desenho.

- Escape (Escape)
Uma facilidade do sistema CORE que é o único acesso para suporte de funções "não-CORE" dependentes de implementação.

5. GKS

A partir do encontro de Seillac, em 1976, vários países se envolveram na tentativa de desenvolvimento de padrões em computação gráfica. O grande desafio dos projetos de padronização nessa área era conceber um sistema gráfico realmente independente de dispositivo que, contudo, permitisse o uso irrestrito das facilidades oferecidas por um determinado dispositivo.

Na Alemanha Ocidental, através do DIN, a entidade nacional de padronização, foi projetado o GKS ("Graphical Kernel System"). Em 1979, a ISO reconheceu o GKS como um ítem de trabalho, e encarregou o seu WG2 ("Working Group 2") de definir um padrão bidimensional, "CORE-like", a partir do projeto do DIN. Em outubro de 1981, foi promovido um encontro em Abingdon, Inglaterra, que resultou na versão 7.0 do GKS, de 14 de janeiro de 1982, registrada na ISO como uma "draft proposal" - ISO DP 7942. Em junho de 1982, em Eidhoven, Holanda, foi realizado o encontro que gerou a versão 7.2 de 14 de novembro de 1982, atualmente um "draft international standard" da ISO.

O ANSI/X3H3, no seu encontro de outubro de 1983, decidiu submeter o GKS à apreciação pública da comunidade norte-americana de Computação Gráfica, com a finalidade de adotá-lo como padrão também nos EUA, berço do sistema CORE. Para isso, publicou em fevereiro de 1984, em edição especial da revista "Computer Graphics", do SIGGRAPH-ACM, a versão do GKS padrão da ISO com ligeiras modificações e com o acréscimo de "binding" para a linguagem FORTRAN, tendo estado aberta ao exame público até 1º de

julho de 1984. No momento, aguarda-se uma decisão final sobre a adoção do GKS como padrão também nos EUA.

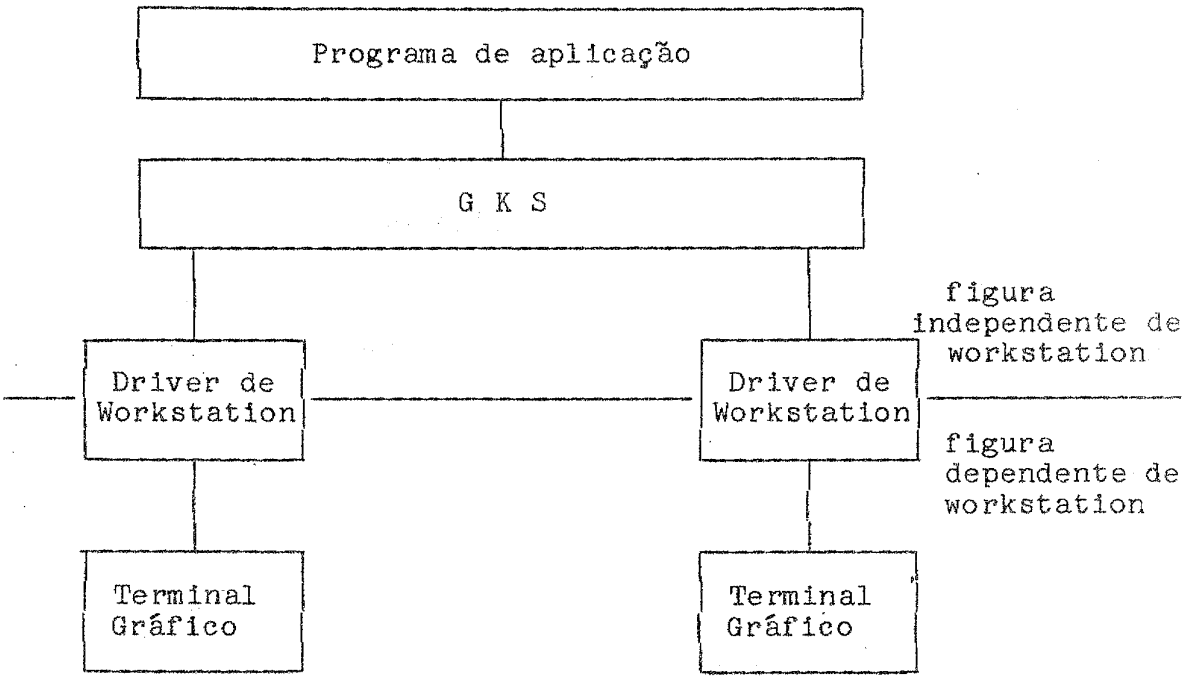
5.1 INDEPENDÊNCIA LÓGICA DE DISPOSITIVO

O GKS é um sistema gráfico padrão bidimensional, que utiliza o conceito de "workstation" (estação de trabalho) como base para se atingir a independência de dispositivo.

A "workstation" suporta um mecanismo de abstração em dois níveis para primitivas de entrada e de saída, especialmente no que diz respeito a seus atributos, incluindo as transformações de coordenadas. Dessa forma, todos os aspectos que podem ser manipulados de forma semelhante nos dispositivos físicos (como, por exemplo, as informações totalmente geométricas) são colocados num nível de abstração mais alto e chamados de aspectos independentes de "workstation". Aspectos com independência de dispositivo limitada, que se aplicam a apenas certas classes de dispositivo (como por exemplo, a cor), estão no nível de abstração mais baixo e são chamados de aspectos dependentes de "workstation".

A "interface" entre esses dois níveis de abstração é realizada no GKS com a definição de um mapeamento também em dois níveis, que leva a duas classes de funções: as que endereçam todas as "workstations" ativas ao mesmo tempo, portanto independentes de "workstation", e as que endereçam uma "workstation" específica identificada por um parâmetro. Dessa

forma, é possível que as "workstations" difiram quanto ao domínio de valores para os atributos que elas suportam. O mapeamento em dois níveis permite que se fixe um atributo primitivo abstrato que é em seguida aplicado no mesmo nível de abstração, ou é mapeado no valor mais apropriado disponível em cada "workstation". O segundo mapeamento também pode ser controlado por funções de "workstation", o que torna possível a duas "workstations" fornecerem representações diferentes para os mesmos atributos. A figura abaixo ilustra a inclusão desse conceito.



Uma outra facilidade básica derivada desse conceito é a capacidade de configurar "workstations" abstratas, já que uma "workstation" é definida como 0 ou 1 dispositivo de entrada e 0 ou mais dispositivos de saída para cada classe lógica de entrada.

A maioria dos dispositivos de entrada depende dos dispositivos de saída disponíveis para fornecer o "feedback" adequado para o operador. Considerando que ecos, "prompts" e outros aspectos de entrada são tratados como atributos dependentes de "workstation", esse problema de dependência de dispositivo é resolvido com a integração na mesma "workstation" de todos os dispositivos de entrada que requerem o mesmo dispositivo de saída.

5.2 A FILOSOFIA DO GKS

Tendo sido desenvolvido sob forte influência da proposta CORE/77, o GKS segue a mesma filosofia emanada do encontro de Seillac, isto é, a busca da portabilidade de programa e de programador. Assim, grande parte de seus conceitos e capacidades funcionais coincidem com os do CORE.

O GKS fornece uma "interface" funcional entre um programa de aplicação e uma configuração de dispositivos gráficos de entrada e saída. A "interface" funcional contém todas as funções básicas para a computação gráfica interativa e passiva, numa ampla variedade de equipamentos gráficos. A "interface" é de alto nível, de forma a proteger o programador de aplicação das peculiaridades do "hardware". Ela oferece primitivas de saída (POLYLINE, POLYMARKER, TEXT, FILL AREA, CELL ARRAY, GENERALIZED DRAWING

PRIMITIVE) e classes de entrada (LOCATOR, STROKE, VALUATOR, CHOICE, PICK, STRING) uniformes.

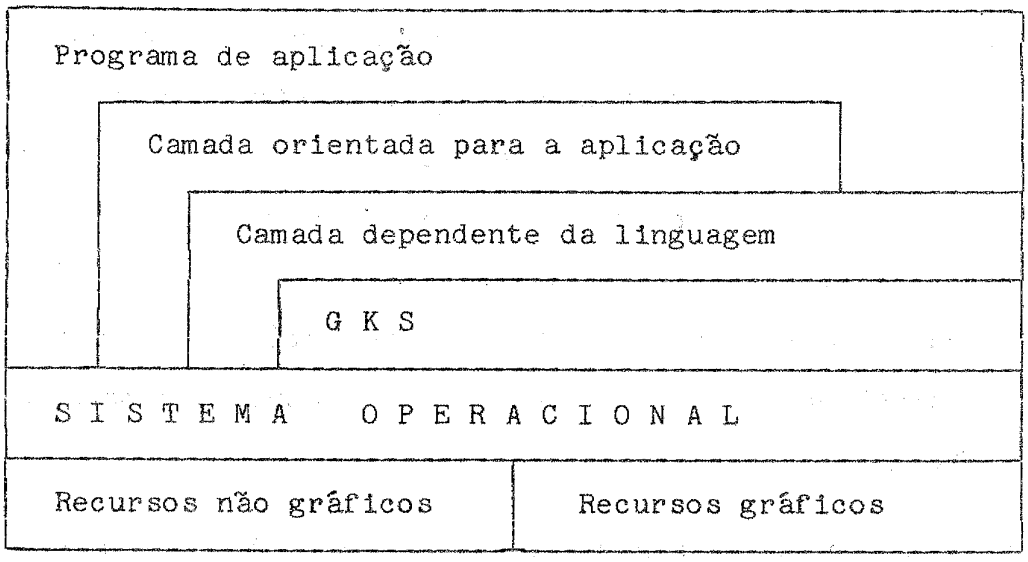
O conceito fundamental para a estruturação do GKS e para a realização da independência de dispositivo é o conceito de "workstation". A possibilidade de definição de várias "workstations" permite saída gráfica e entrada simultâneas em vários sistemas de exibição. São fornecidas facilidades para armazenamento interno e externo, através de "workstations" especiais, bem como a possibilidade de transferência de entidades gráficas diretamente dessas "workstations" especiais para outras "workstations".

Facilidades para manipulação e alteração de figuras são introduzidas pelos conceitos de segmentação, atributos dinâmicos e transformações de imagem.

O sistema foi projetado em nove níveis de implementação (doze na versão publicada pelo ANSI), de forma a atingir as diferentes necessidades das aplicações gráficas. Cada implementação do GKS deve fornecer, pelo menos, as funções de um desses níveis. Os níveis superiores abrangem os inferiores.

O GKS constitui apenas um núcleo de um sistema gráfico independente de linguagem. Para integração com a linguagem, está embutido numa camada dependente da linguagem que contém as regras de "binding", como por exemplo, atribuição de nomes para as rotinas e passagem de parâmetros. A versão publicada pelo ANSI contém uma especificação dessas regras para a linguagem FORTRAN.

O modelo de camadas da figura a seguir ilustra o papel do GKS num sistema gráfico.



Cada camada pode chamar as funções das camadas inferiores adjacentes. Em geral, o programa de aplicação usa a camada orientada para a aplicação, a camada dependente da linguagem, outras camadas dependentes de aplicação e recursos do sistema operacional.

5.3 A ESPECIFICAÇÃO FUNCIONAL DO GKS

Os conceitos e especificações apresentados a seguir se referem à versão publicada pelo ANSI que, na sua essência, são os mesmos da versão padronizada pela ISO.

Os conceitos que, por serem os mesmos, já foram definidos na descrição do CORE, são omitidos na descrição da especificação funcional do GKS.

5.3.1 Primitivas de Saída

O GKS fornece seis primitivas de saída :

POLYLINE - gera um conjunto de linhas retas ligando uma dada sequência de pontos.

POLYMARKER - gera símbolos do mesmo tipo centrados em posições dadas.

TEXT - gera uma cadeia de caracteres numa dada posição.

FILL AREA - gera um polígono que pode ser vazio ou preenchido com uma cor, um padrão ou um estilo de hachuras.

CELL ARRAY - gera uma matriz de células retangulares com cores individuais; é uma generalização da matriz de "pixels" num dispositivo "raster".

GENERALIZED DRAWING PRIMITIVES - endereça capacidades geométricas especiais de saída de uma "workstation" como o desenho de curvas.

Como o GKS não utiliza o conceito de posição corrente, os pontos iniciais dos desenhos devem ser especificados explicitamente. O traçado de uma única linha ou a marcação de um único ponto são casos particulares das primitivas POLYLINE e POLYMARKER.

As primitivas FILL AREA e CELL ARRAY são fornecidas para

serem usadas em "workstations" com dispositivos de saída tipo "raster".

São mostradas a seguir as formas de chamada das funções primitivas de saída:

```
POLYLINE (n,points)
POLYMARKER (n,points)
TEXT (position,string)
FILL_AREA (n,points)
CELL_ARRAY (cell_retangle,dimension,colour_index_array)
GENERALIZED_DRAWING_PRIMITIVES (n,points,GDP_identifier,
                                GDP_data_record)
```

5.3.2 Segmentação

As primitivas gráficas podem ser grupadas em segmentos ou podem ser criadas fora de segmentos. O segmento é a unidade de manipulação e alteração da figura. Manipulação inclui criação, exclusão e mudança de nome. Alteração inclui transformar a imagem de um segmento, tornar um segmento invisível e aumentar a intensidade ("highlighting") de um segmento. Apenas as primitivas contidas dentro de segmentos podem ser afetadas pelas operações de segmentos acima; o programa de aplicação não tem acesso a primitivas fora de segmentos após a sua criação.

Os segmentos são também a unidade de armazenamento de figuras independentemente de "workstations", o que possibilita a inserção e a transferência de segmentos entre "workstations".

As funções de segmentação do GKS são as seguintes:

```
CREATE_SEGMENT (segment_name)
CLOSE_SEGMENT
RENAME_SEGMENT (old_segment_name,new_segment_name)
DELETE_SEGMENT (segment_name)
DELETE_SEGMENT_FROM_WORKSTATION (ws_identifier,seg_name)
ASSOCIATE_SEGMENT_WITH_WORKSTATION (ws_identifier,seg_name)
COPY_SEGMENT_TO_WORKSTATION (ws_identifier,seg_name)
INSERT_SEGMENT (segment_name,transformation_matrix)
```

5.3.3 Atributos

Os atributos de primitivas controlam os aspectos das primitivas de saída num dispositivo. Os atributos são classificados em três tipos: geométricos, não-geométricos e de identificação ("pick identifier"). Os atributos de primitivas são estáticos, isto é, não podem ser alterados após a criação das primitivas.

Os atributos geométricos são estabelecidos numa forma independente de "workstation". Os atributos não geométricos são aqueles que não afetam a forma ou o tamanho de uma primitiva mas apenas a sua aparência. Podem ser controlados de duas formas: individualmente ou em grupo ("bundled") através de um índice que aponta para uma tabela que contém combinações de valores para os atributos não geométricos da primitiva. O sistema permite ao programa de aplicação estabelecer a forma (individual ou "bundled") de controle de cada atributo, através de um "flag", e permite, também, criar diferentes representações dos grupos de atributos não geométricos para cada "workstation".

A tabela abaixo mostra os atributos de cada primitiva de saída, de acordo com o seu tipo.

Primitiva	Atributos geométricos	Atributos não geométricos
POLYLINE		LINETYPE LINEWIDTH SCALE FACTOR COLOUR
POLYMARKER		MARKERTYPE MARKER SIZE SCALE FACTOR COLOUR
TEXT	CHARACTER HEIGHT CHARACTER UP VECTOR TEXT PATH TEXT ALIGNMENT	CHARACTER EXPANSION FACTOR CHARACTER SPACING TEXT FONT AND PRECISION TEXT COLOUR INDEX
FILL AREA	PATTERN SIZE PATTERN REFERENCE POINT	PATTERN ARRAY INTERIOR STYLE HATCH STYLE COLOUR

Observações :

- Todas as primitivas possuem o atributo de identificação PICK IDENTIFIER.
- A primitiva CELL ARRAY só possui o atributo PICK IDENTIFIER.
- A primitiva GENERALIZED DRAWING PRIMITIVES possui, além do PICK IDENTIFIER, atributos dependentes das primitivas utilizadas na função.

As funções que controlam atributos de primitivas estão listadas a seguir.

```
SET_POLYLINE_INDEX (index)
SET_LINETYPE (type)
SET_LINEWIDTH_SCALE_FACTOR (factor)
SET_POLYLINE_COLOUR_INDEX (index)
```

```
SET_POLYMARKER_INDEX (index)
SET_MARKER_TYPE (type)
SET_MARKER_SIZE_SCALE_FACTOR (factor)
SET_POLYMARKER_COLOUR_INDEX (index)

SET_TEXT_INDEX (index)
SET_FONT_AND_PRECISION (font,precision)
SET_CHARACTER_EXPANSION_FACTOR (factor)
SET_CHARACTER_SPACING (spacing)
SET_TEXT_COLOUR_INDEX (index)
SET_CHARACTER_HEIGHT (height)
SET_CHARACTER_UP_VECTOR (up_vector)
SET_TEXT_PATH (path)
SET_TEXT_ALIGNMENT (alignment)

SET_FILL_AREA_INDEX (index)
SET_FILL_AREA_INTERIOR_STYLE (style)
SET_FILL_AREA_STYLE_INDEX (index)
SET_FILL_AREA_COLOUR_INDEX (index)

SET_PATTERN_SIZE (size)
SET_PATTERN_REFERENCE_POINT (point)

SET_ASPECT_FLAGS (list_of_flags)

SET_PICK_IDENTIFIER (identifier)

SET_POLYLINE_REPRESENTATION (ws_identifier,index,linetype,
          linewidth_scale_factor,colour_index)
SET_POLYMARKER_REPRESENTATION (ws_identifier,index,
          markertype,marker_size_scale_factor,colour_index)
SET_TEXT_REPRESENTATION (ws_identifier,index,font,precision,
          char_expansion_factor,char_spacing,colour_index)
SET_FILL_AREA_REPRESENTATION (ws_identifier,index,
          interior_style,style_index,colour_index)
SET_PATTERN_REPRESENTATION (ws_identifier,index,dimension,
          pattern_array)
SET_COLOUR_REPRESENTATION (ws_identifier,index,colour)
```

A aparência dos segmentos é controlada por atributos de segmentos que são: visibilidade, "highlighting", detectabilidade, prioridade de segmento e transformação de segmento. Esses atributos podem ser estabelecidos dinamicamente.

A prioridade de segmentos afeta somente os segmentos que estão sendo exibidos. Se partes de primitivas se sobrepõem com outras de um segmento visível de prioridade maior, essas partes

são tornadas invisíveis. Quando primitivas dentro de segmentos que se sobrepõem são detectadas por "pick", o segmento de maior prioridade será selecionado.

São as seguintes as funções que controlam atributos de segmentos:

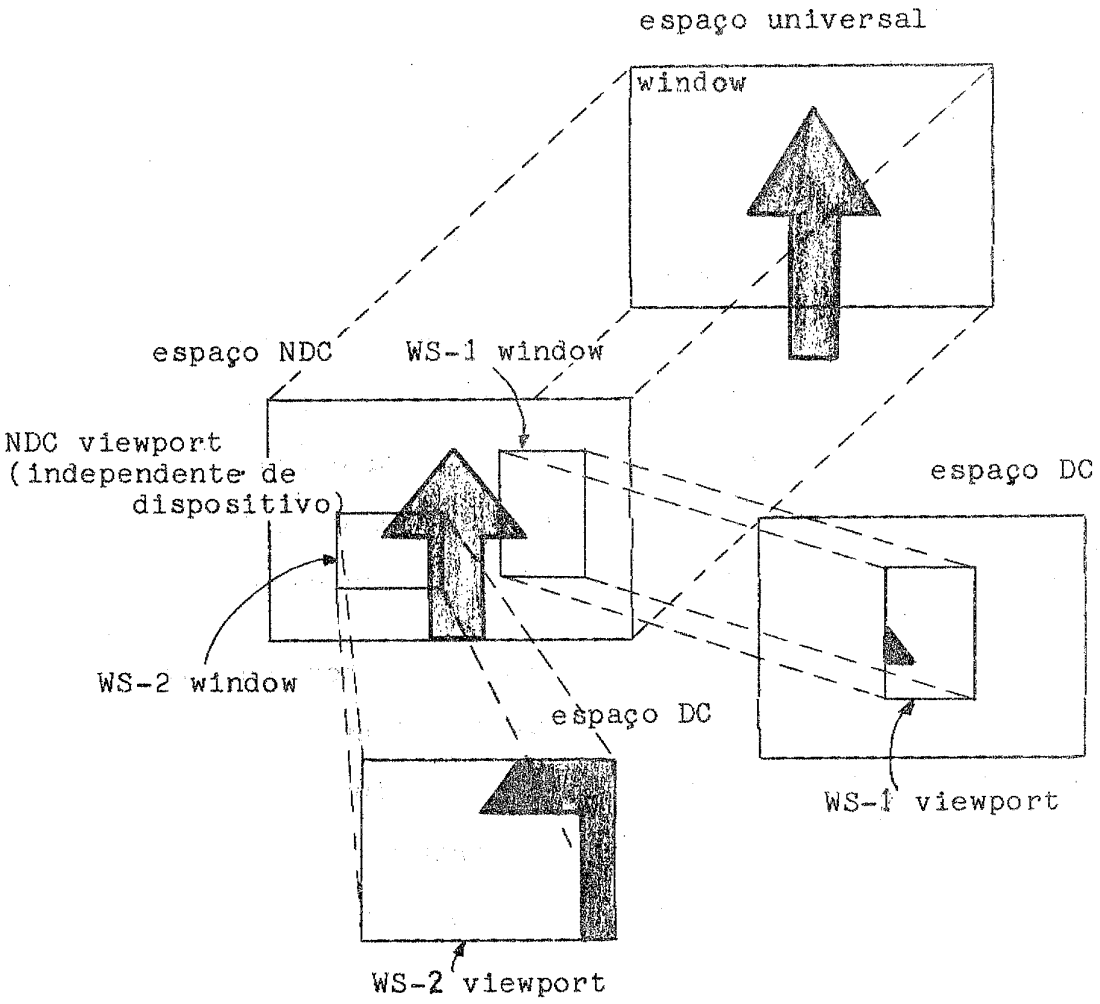
```
SET_SEGMENT_TRANSFORMATION (seg_name,transformation_matrix)
SET_DETECTABILITY (segment_name,detectability)
SET_VISIBILITY (segment_name,visibility)
SET_HIGHLIGHTING (segment_name,highlighting)
SET_SEGMENT_PRIORITY (segment_name,priority)
```

5.3.4 Funções de Transformação

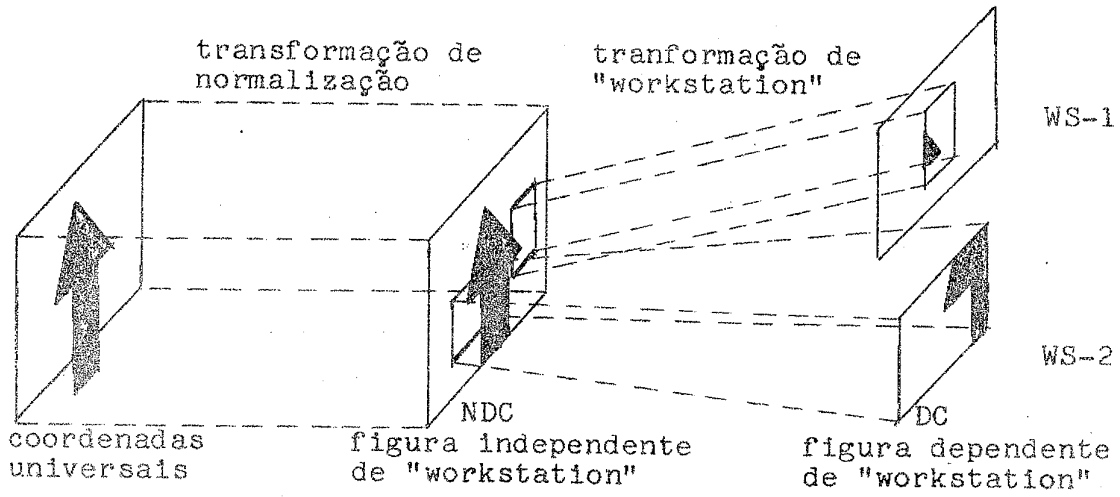
O GKS permite que o programador defina um espaço de coordenadas, chamado espaço de coordenadas universais, que é apropriado para cada aplicação. Esse espaço é mapeado para coordenadas do dispositivo de forma controlada, através de duas operações: transformações de normalização e transformações de "workstation". Inicialmente, o GKS transforma coordenadas universais no espaço de coordenadas normalizadas do dispositivo (espaço NDC) através da definição de uma "window" (janela), no espaço de coordenadas universais. O espaço NDC é então transformado em coordenadas de dispositivo (DC) da estação. Quando são usadas várias "workstations", cada uma pode ter uma "viewport" (vista) diferente da aplicação através da definição de sua "worskstation window". A última transformação permite que a "workstation" defina uma "viewport", a região ativa do espaço de trabalho potencial do dispositivo, a qual pode ser usada para mudança de

.a e translação da figura original.

A figura abaixo ilustra esses conceitos.



A figura seguinte situa as transformações.



Cada "window" e cada "viewport" da transformação de normalização são identificadas pelo número da transformação a que estão associadas, de modo que esta pode ser selecionada a qualquer tempo no decorrer do programa. Na estrutura de um programa de aplicação que usa o GKS, costuma-se declarar no início todas as transformações que serão utilizadas no corpo do programa.

Cada transformação de normalização tem associada uma prioridade de entrada VIEWPORT INPUT PRIORITY, que estabelece a transformação de normalização cuja inversa deve ser usada no mapeamento de coordenadas do dispositivo para coordenadas universais, quando ocorrer uma entrada a partir de um dispositivo da classe LOCATOR ou STROKE.

As seguintes funções controlam as transformações no GKS:

```
SET_WINDOW (transf_number,xmin,xmax,ymin,ymax)
SET_VIEWPORT (transf_number,xmin,xmax,ymin,ymax)
SET_VIEWPORT_INPUT_PRIORITY (transf_number,
                        reference_transf_number,priority)
SET_NORMALIZATION_TRANSFORMATION (transf_number)
SET_CLIPPING_INDICATOR (indicator)
SET_WORKSTATION_WINDOW (ws_identifier,xmin,xmax,ymin,ymax)
SET_WORKSTATION_VIEWPORT (ws_identifier,xmin,xmax,ymin,ymax)
```

5.3.5 Primitivas de Entrada

Um amplo conjunto de operações de entrada permite que um programa de aplicação receba entrada gráfica de uma grande variedade de dispositivos. As operações de entrada são grupadas em seis classes lógicas: LOCATOR, STROKE, VALUATOR, CHOICE, PICK e STRING.

Das classes de dispositivos lógicos de entrada, somente o LOCATOR, o STROKE e o PICK podem ser chamados de verdadeiramente gráficos. Contudo, para o projeto de uma "interface" poderosa de comunicação interativa entre um programa de aplicação e um operador, todas essas facilidades são necessárias.

Cada dispositivo lógico de entrada pode ser operado em um dos três diferentes modos de operação :

REQUEST - a entrada é produzida pelo operador como resposta a uma operação de "leitura" do programa de aplicação.

SAMPLE - o valor corrente de entrada de um dispositivo especificado é obtido diretamente pelo programa de aplicação, sem esperar por uma ação do operador.

EVENT - a entrada é gerada assincronamente pelo operador e é coletada numa fila de eventos para o programa de aplicação.

Mecanismos de eco e "prompt" são estabelecidos pela função de inicialização de cada um dos dispositivos lógicos de entrada.

As funções do GKS que manipulam as operações de entrada são

resumidas a seguir.

INITIALISE dispositivo lógico (lista de parâmetros)

SET dispositivo lógico MODE (ws_identifier,device_number,
operating_mode,echo_switch)

REQUEST dispositivo lógico (lista de parâmetros)

SAMPLE dispositivo lógico (lista de parâmetros)

AWAIT_EVENT (time_out,ws_identifier,input_class)

FLUSH_DEVICE_EVENTS (ws_identifier,input_class,
device_number)

GET dispositivo lógico (lista de parâmetros)

5.3.6 Funções de Controle e de Consulta

O GKS possui funções de controle que executam operações como abrir e fechar o sistema, abrir e fechar "workstations", ativar e desativar "workstations". A ação de "new frame" é executada pela função CLEAR WORKSTATION e a rotina que exibe a figura corrente de uma "workstation" é o UPDATE WORKSTATION.

O controle do tempo das alterações na figura é executado pelo estabelecimento de "deferral states" (estados de adiamento) que podem ser dos seguintes tipos, em ordem crescente de adiamento :

ASAP ... "As Soon As Possible"

BNIG ... "Before the Next Interaction Globally"

BNIL ... "Before the Next Interaction Locally"

ASTI ... "At Some Time"

São as seguintes as funções de controle do GKS:

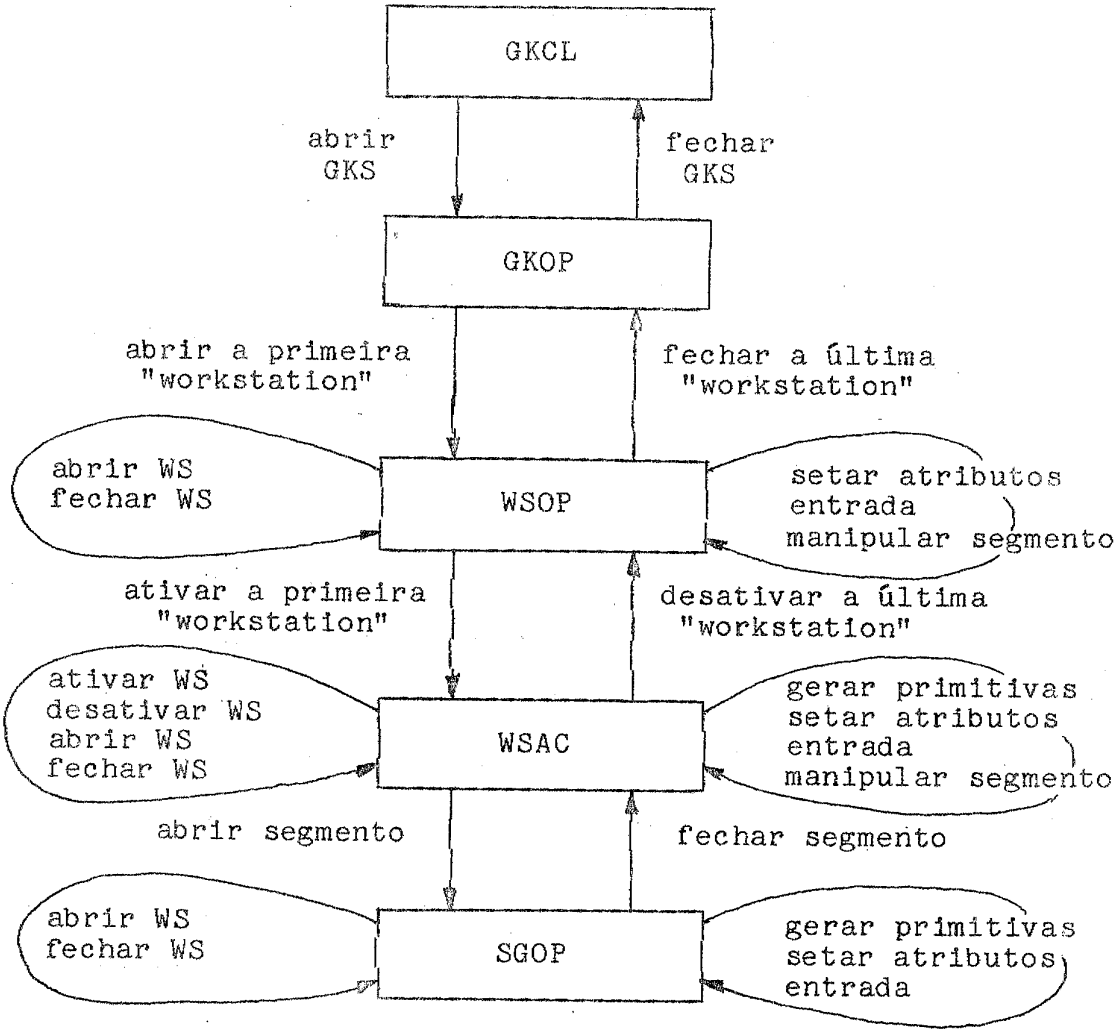
```
OPEN_GKS (error_file,buffer_area)
CLOSE_GKS
OPEN_WORKSTATION (ws_identifier,connection_identifier,type)
CLOSE_WORKSTATION (ws_identifier)
ACTIVATE_WORKSTATION (ws_identifier)
DEACTIVATE_WORKSTATION (ws_identifier)
CLEAR_WORKSTATION (ws_identifier,control_flag)
REDRAW_ALL_SEGMENTS_ON_WORKSTATION (ws_identifier)
UPDATE_WORKSTATION (ws_identifier,regeneration_flag)
SET_DEFERRAL_STATE (ws_identifier,deferral_mode,
                    regeneration_mode)
MESSAGE (ws_identifier,message)
ESCAPE (escape_function_ident,escape_data_record)
```

Como auxílio ao programador, o GKS fornece uma capacidade de consulta (INQUIRY) que permite ao programa de aplicação obter uma vasta gama de informações acerca do ambiente do seu sistema: o estado operacional corrente, atributos, operações de visualização e de transformação, e capacidades das "workstations".

A qualquer tempo, durante a execução de um programa que usa o GKS, este deve estar num estado operacional precisamente definido. São cinco os estados operacionais do GKS :

- GKS fechado (GKCL);
- GKS aberto (GKOP);
- Pelo menos uma "workstation" aberta (WSOP);
- Pelo menos uma "workstation" ativa (WSAC);
- Segmento aberto (SGOP).

A figura seguinte ilustra as transições entre os estados operacionais e as funções permitidas em cada estado.



5.3.7 "Interface" para Metafile

O GKS fornece uma "interface" para arquivos sequenciais com o propósito de armazenamento de informação gráfica, chamada "GKS metafile" (GKSM). O "metafile" pode ser usado para :

- transporte de informação gráfica entre sistemas;
- transporte de informação gráfica de uma instalação para outra (por exemplo, por meio de fita magnética);

- transporte de informação gráfica de uma aplicação para outra;
- armazenamento de informações não gráficas associadas à aplicação.

As funções padronizadas para "interface" do programa de aplicação com o "metafile" são:

```
WRITE_ITEM_TO_GKSM (ws_identifier,type,length,data_record)
GET_ITEM_TYPE_FROM_GKSM (ws_identifier,type,length)
READ_ITEM_FROM_GKSM (ws_identifier,max_length,data_record)
INTERPRET_ITEM (type,length,data_record)
```

5.3.8 Outras Funções

O GKS provê funções utilitárias, que fazem a montagem da matriz de transformação e permitem a composição de transformações, bem como funções para tratamento de erros do programa de aplicação.

Funções utilitárias

```
EVALUATE_TRANSFORMATION_MATRIX (fixed_point,shift_vector,
        rotation_angle,scale_factors,coordinate_switch)

ACCUMULATE_TRANSFORMATION_MATRIX (segment_transf_matrix,
    fixed_point,shift_vector,rotation_angle,scale_factors,
    coordinate_switch)
```

Funções de manipulação de erro

```
EMERGENCY_CLOSE_GKS

ERROR_HANDLING (error_number,procedure_ident,error_file)
```

5.4 NÍVEIS DE IMPLEMENTAÇÃO DO SISTEMA

A fim de possibilitar o uso do sistema numa ampla variedade de aplicações de uma maneira uniforme, o GKS é estruturado em níveis de implementação, com diferentes capacidades funcionais.

As capacidades funcionais do GKS são grupadas nas seguintes áreas principais:

- saída (desempenho mínimo até completo);
- entrada (nenhuma entrada, entrada REQUEST, entrada completa);
- número de "workstations" (uma "workstation", múltiplas "workstations");
- atributos (somente "bundles" pré-definidos e especificações individuais de atributos, "bundle" completo);
- segmentação (nenhuma, segmentação básica - sem o armazenamento de segmentos independente de "workstation", segmentação completa).

A estrutura de níveis do GKS tem dois eixos independentes : entrada e todas as demais funções, resumidas como saída.

O eixo do nível de saída tem quatro possibilidades (três na versão padronizada pela ISO):

m : saída mínima;

0 : todas as primitivas e atributos;

1 : segmentação básica com saída completa;

2 : armazenamento de segmentos independente de "workstation".

O eixo do nível de entrada tem três possibilidades:

a : nenhuma entrada;

b : entrada no modo REQUEST;

c : entrada completa.

A tabela abaixo dá uma visão da funcionalidade de cada nível GKS válido. Cada caixa contém somente as funções adicionadas às caixas anteriores da mesma linha e coluna.

Nível de Saída	Nível de Entrada		
	a	b	c
m	nenhuma entrada, controle mínimo, somente atributos individuais, uma única transformação de normalização, subconjunto de funções de saída e atributos.	entrada REQUEST, setar modos, inicializar dispositivos, sem PICK	entrada SAMPLE e EVENT, sem PICK
0	controle básico, "bundles" pré-definidos, múltiplas transformações de normalização, no mínimo 1, todas as funções de saída, "metafile workstation" opcional	prioridade de entrada de "viewport"	
1	saída completa com "bundle" total, "workstations" múltiplas, segmentação básica, "metafile workstations"	setar modos e inicializar PICK em REQUEST	entrada SAMPLE e EVENT para PICK
2	armazenamento de segmentos independente de "workstation"		

5.5 "BINDINGS" PARA LINGUAGENS DE PROGRAMAÇÃO

A versão ANS GKS apresenta uma seção de padronização de "bindings" das funções GKS para linguagens de programação, em cooperação com os comitês de linguagens. Em particular, apresenta um padrão de "binding" para o FORTRAN/77.

5.6 GLOSSÁRIO

Este é o glossário da versão ANS GKS, idêntico ao da versão ISO. Foi mantida a ordem alfabética em inglês e as traduções utilizadas no texto estão entre parênteses. Quando a tradução não foi possível, manteve-se o termo original.

- Acknowledgement (Conhecimento)
  Saída para o operador de um dispositivo de entrada lógico, indicando que foi disparado um "trigger".

- Aspects of Primitives (Aspectos de Primitivas)
  Formas de variação da aparência de uma primitiva. Alguns aspectos são controlados diretamente pelos atributos de primitivas, outros são controlados indiretamente através de uma tabela "bundle". Primitivas dentro de segmentos tem seu aspecto controlado através do segmento que as contém, por exemplo, "highlighting", o que não ocorre com primitivas fora de segmentos.

- Attribute (Atributos)
  Propriedade específica que se aplica a um elemento de exibição (primitiva de saída) ou um segmento. No GKS, algumas propriedades de "workstations" são chamadas de atributos de "workstations".

- Bundle Index (Índice "bundle")
  Índice numa tabela "bundle" para uma primitiva de saída específica.

- Bundle Table (Tabela "bundle")
  Tabela dependente de "workstation" associada a uma primitiva de saída específica. As entradas na tabela especificam todos os aspectos dependentes de "workstation" de uma primitiva. No GKS, existem tabelas "bundle" para as seguintes primitivas de saída: POLYLINE, POLYMARKER, TEXT e FILL AREA.

- Cell Array (Matriz de células)
  Primitiva de saída GKS que consiste de uma grade retangular de células retangulares de mesmo tamanho, cada uma tendo uma cor. Não existe, necessariamente, um mapeamento 1:1 dessas células para "pixels".

- Choice Device (Dispositivo de Escolha)
  Dispositivo de entrada lógico do GKS que fornece um número inteiro não negativo que define uma entre várias alternativas.

- Clipping (Cerceamento)
  Remoção de partes de elementos de exibição que se localizam

fora de uma dada fronteira, usualmente uma "window" ou uma "viewport".

- Colour Table (Tabela de Cores)
Tabela dependente de "workstation", na qual as entradas especificam os valores das intensidades vermelho, verde e azul que definem uma determinada cor.

- Coordinate Graphics; Line Graphics (Computação Gráfica de Coordenadas ou de Linhas)
Computação gráfica na qual imagens são geradas a partir de comandos de exibição e dados em coordenadas.

- Device Coordinate (DC) (Coordenada de Dispositivo)
Coordenada expressa num sistema de coordenadas que é dependente de dispositivo. No GKS, unidades DC são metros em dispositivos capazes de produzir variações precisas na escala da imagem; caso contrário, as unidades são as apropriadas para cada "workstation".

- Device Driver ("Driver" de dispositivo)
A parte da implementação do GKS dependente de dispositivo que suporta um dispositivo gráfico. O "driver" de dispositivo gera saída dependente de dispositivo e manipula interação dependente de dispositivo.

- Device Space (Espaço de Dispositivo)
O espaço definido pelos pontos endereçáveis de uma superfície de exibição.

- Display Device; Graphics Device (Dispositivo de Exibição, Dispositivo Gráfico)
Um dispositivo no qual imagens de exibição podem ser representadas. (por exemplos, "storage tube display", "plotter")

- Display Element (Elemento de Exibição)
Elemento gráfico básico que pode ser usado para construir uma imagem de exibição.

- Display Image; Picture (Imagem de Exibição, Figura)
Coleção de elementos de exibição ou segmentos que são representados juntos na superfície de exibição a qualquer hora.

- Display Space (Espaço de Exibição)
A porção do espaço de dispositivo correspondente à área disponível para exibir imagens. No GKS, o espaço de exibição é utilizado também para se referir ao espaço de trabalho de um dispositivo de entrada, como um digitalizador.

- Display Surface; View Surface (Superfície de Exibição; Superfície de Visualização)
Num dispositivo de exibição, o meio no qual as imagens podem aparecer.

- Echo (Eco)
A notificação imediata do valor corrente fornecido por um dispositivo de entrada ao operador na "console" de exibição.

- Escape (Escape)
Uma função GKS usada para acessar facilidades dependentes de implementação ou de dispositivo, que não sejam para geração de saída gráfica e que não sejam fornecidas pelo GKS.

- Feedback (Realimentação)
Saída que indica ao operador a interpretação do programa de aplicação de um valor de entrada lógico.

- Fill Area
Primitiva de saída do GKS que consiste de um polígono fechado que pode estar vazio ou preenchido por uma cor uniforme, um padrão ou um estilo de hachuras.

- Fill Area Bundle Table (Tabela "Bundle" de "Fill Area")
Tabela que associa valores específicos para todos os aspectos dependentes de "workstation" de uma primitiva "fill area" com um índice "bundle" de "fill area". No GKS, essa tabela contém entradas que consistem de estilo de interior, índice de estilo e índice de cor.

- Generalized Drawing Primitive (GDP) (Primitiva de Desenho Generalizada)
Primitiva de saída usada para endereçar capacidades geométricas especiais de "workstation" como desenho de curvas.

- GKS Level (Nível GKS)
Dois valores dos intervalos (m,0,1,2) e (a,b,c) que juntos definem as capacidades funcionais mínimas fornecidas por uma implementação específica do GKS.

- GKS Metafile (GKSM)
Um arquivo sequencial que pode ser escrito ou lido pelo GKS; utilizado para armazenamento por longo tempo (e para transmissão e transferência) de informações gráficas.

- Highlighting
Uma forma dependente de dispositivo de enfatizar um segmento pela modificação de seus atributos visuais. (uma generalização do "blinking")

- Implementation Mandatory (Obrigatório na Implementação)
Descreve uma propriedade que deve ser realizada de forma idêntica em todas as "workstations" de todas as implementações do padrão.

- Input Class (Classe de Entrada)
Conjunto de dispositivos de entrada que são logicamente equivalentes com respeito à sua função. No GKS, as classes de entrada são: LOCATOR, STROKE, VALUATOR, CHOICE, PICK e STRING.

- Locator Device (Dispositivo "Locator")
Dispositivo lógico de entrada que fornece uma posição em coordenadas universais e um número de transformação de normalização.

- Logical Input Device (Dispositivo de Entrada Lógico)
É uma abstração de um ou mais dispositivos físicos que fornece valores de entrada ao programa. Dispositivos de entrada lógicos correspondem às classes de entrada.

- Logical Input Value (Valor de Entrada Lógico)
Valor fornecido por um dispositivo de entrada lógico.

- Marker (Marcador)
Um símbolo com uma aparência específica que é usado para identificar uma posição particular.

- Measure (Medida)
Valor (associado com um dispositivo de entrada lógico), que é determinado por um ou mais dispositivos de entrada físicos e um mapeamento dos valores fornecidos pelos dispositivos físicos. O valor de entrada lógico fornecido por um dispositivo de entrada lógico é o valor corrente da medida.

- Normalization Transformation; Viewing Transformation; Window-to-Viewport Transformation (Transformação de Normalização, Transformação de Visualização, Transformação Janela-Vista)
Transformação que mapeia o contorno e o interior de uma janela para o contorno e o interior de uma vista. No GKS, esta transformação mapeia posições em coordenadas universais para coordenadas normalizadas de dispositivo.

- Normalized Device Coordinates (NDC) (Coordenadas de Dispositivo Normalizadas)
Coordenadas especificadas num sistema de coordenadas intermediário independente de dispositivo, normalizado num certo intervalo, tipicamente 0 a 1. No GKS, durante um estado intermediário, as coordenadas podem se situar fora do intervalo definido, mas a informação de cerceamento associada assegura que a saída não excederá o domínio das coordenadas (0,1)x(0,1).

- Operator (Operador)
Pessoa que manipula dispositivos físicos de entrada de forma a alterar as medidas dos dispositivos lógicos de entrada e causar o acionamento dos seus "triggers".

- Output Primitive; Graphic Primitive (Primitiva de Saída; Primitiva Gráfica)
Elemento de exibição. As primitivas de saída no GKS são: POLYLINE, POLYMARKER, TEXT, FILL AREA, CELL ARRAY e GENERALIZED DRAWING PRIMITIVE.

- Pick Device (Dispositivo de "Pick")
Dispositivo de entrada lógico que fornece o identificador de "pick" associado a uma primitiva de saída e o nome do segmento

que a contém.

- Pixel; Picture Element (Pixel, Elemento de Figura)
O menor elemento de uma superfície de exibição ao qual pode ser independentemente atribuída uma cor ou uma intensidade.

- Polyline
Primitiva de saída GKS que consiste de um conjunto de linhas ligadas.

- Polyline Bundle Table (Tabela "Bundle" de Polyline)
Tabela que associa valores específicos a todos os aspectos dependentes de "workstation" de primitivas "polyline" com um índice "bundle" de "polyline". No GKS, essa tabela contém entradas que consistem de tipo de linha, fator de escala de largura de linha e índice de cor.

- Polymarker
Primitiva de saída GKS que consiste de um conjunto de posições, todas indicadas pelo mesmo tipo de marcador.

- Polymarker Bundle Table (Tabela "Bundle" de "Polymarker")
Tabela que associa valores específicos para todos os aspectos dependentes de "workstation" de uma primitiva "polymarker" com um índice "bundle" de polymarker. No GKS, essa tabela contém entradas que consistem de um tipo de marcador, fator de escala de tamanho de marcador e índice de cor.

- Primitive Attribute (Atributo de Primitiva)
Atributo que se aplica a primitivas de saída, diferente de atributos que se aplicam a outros aspectos do sistema gráfico como segmentos. Valores de atributos de primitivas (para primitivas de saída) são selecionados pela aplicação de uma forma independente de "workstation", mas podem ter efeitos dependentes de "workstation".

- Prompt
Saída para o operador indicando que um dispositivo de entrada lógico específico está disponível.

- Raster Graphics (Computação Gráfica "Raster")
Computação Gráfica em que a imagem de exibição é composta de um "array" de "pixels" dispostos em linhas e colunas.

- Rotation (Rotação)
Girar toda ou parte da imagem de exibição em torno de um eixo. No GKS, essa capacidade é restrita a segmentos.

- Scaling; Zooming (Mudança de escala)
Aumentar ou reduzir toda ou parte da imagem de exibição multiplicando as coordenadas dos elementos de exibição por um valor constante. Para "scaling" diferente em duas direções ortogonais, são necessários dois valores constantes. No GKS, essa capacidade é restrita a segmentos.

- Segment (Segmento)
  Coleção de elementos de exibição que pode ser manipulada como uma unidade.

- Segment Attributes (Atributos de Segmento)
  Atributos que se aplicam somente a segmentos. No GKS, os atributos de segmentos são : visibilidade, "highlighting", detectabilidade, prioridade de segmento e transformação de segmento.

- Segment Priority (Prioridade de Segmentos)
  Atributo de segmento usado para determinar qual dentre vários segmentos sobrepostos tem precedência para saída gráfica e entrada.

- Segment Transformation (Transformação de Segmento)
  Transformação que faz com que elementos de exibição definidos por um segmento apareçam em variadas posições (translação), tamanho (mudança de escala) e/ou orientação (rotação) na superfície de exibição.

- String Device (Dispositivo "String")
  Dispositivo de entrada lógico do GKS que fornece uma cadeia de caracteres como seu resultado.

- Stroke Device (Dispositivo "Stroke")
  Dispositivo de entrada lógico do GKS que fornece uma seqüência de pontos em coordenadas universais e um número de transformação de normalização.

- Text (Texto)
  Primitiva de saída do GKS que consiste de uma cadeia de caracteres.

- Text Bundle Table (Tabela "Bundle" de Texto)
  Tabela que associa valores específicos de todos os aspectos dependentes de "workstation" de primitivas "text" com um índice "bundle" de "text". No GKS, essa tabela contém entradas consistindo de precisão e "font" de "text", fator de expansão de caracter, espaçamento de caracter e índice de cor.

- Text Font and Precision ("Font" e Precisão de Texto)
  Aspecto de texto no GKS com duas componentes, "font" e precisão, que juntas determinam a forma dos caracteres sendo exibidos numa determinada "workstation". Além disso, a precisão descreve a fidelidade com a qual os outros aspectos de texto se igualam com aqueles requisitados por um programa de aplicação. Em ordem crescente de fidelidade, as precisões são: STRING, CHAR e STROKE.

- Translation; Shift (Translação)
  Aplicação de um deslocamento constante na posição de toda ou de parte de uma imagem de exibição. No GKS, essa capacidade é restrita a segmentos.

- Trigger
Dispositivo ou conjunto de dispositivos de entrada físicos que podem ser usados por um operador para indicar momentos significativos no tempo.

- Valuator Device (Dispositivo "Valuator")
Dispositivo de entrada lógico que fornece um número real.

- Viewport (Vista)
Parte do espaço de coordenadas de dispositivo normalizadas especificada pelo programa de aplicação. No GKS, esta definição é restrita à região retangular do espaço de coordenadas de dispositivo normalizadas utilizada na transformação de normalização.

- Window (Janela)
Parte pré-definida de um espaço virtual. No GKS, esta definição é restrita à região retangular do espaço de coordenadas universais usada para a definição de transformação de normalização.

- Workstation (Estação de Trabalho)
O GKS é baseado no conceito de "workstations" gráficas abstratas, que fornecem a "interface" lógica através da qual o programa de aplicação controla dispositivos físicos.

- Workstation Mandatory (Obrigatório na "Workstation")
Descreve uma propriedade que deve ser realizada de forma idêntica em todas as "workstations" de uma implementação padrão.

- Workstation Transformation (Transformação de "Workstation")
Transformação que mapeia o contorno e o interior de uma janela de "workstation" para o contorno e o interior de uma vista de "workstation" (parte do espaço de exibição), preservando a razão de aspecto. No GKS, essa transformação mapeia posições em coordenadas de dispositivo normalizadas para coordenadas de dispositivo. O efeito de preservação da razão de aspecto é que o interior da janela de "workstation" não pode ser mapeado para a totalidade da vista da "workstation".

- Workstation Viewport (Vista de "Workstation")
Porção do espaço de exibição correntemente selecionada para saída gráfica.

- Workstation Window (Janela de "Workstation")
Região retangular dentro do sistema de coordenadas de dispositivo normalizadas que é representada num espaço de exibição.

- World Coordinate (WC) (Coordenadas Universais)
Sistema de coordenadas cartesianas independente de dispositivo usado pelo programa de aplicação para especificar entrada e saída gráfica.

6. CORE X GKS

As duas diferenças básicas entre os padrões CORE e GKS são:

- a dimensão, que no CORE é 3D e no GKS é 2D;
- e a inclusão de um nível de abstração na estrutura do sistema através do conceito de "workstation" no GKS.

A definição de um padrão 3D é uma tarefa mais complexa e a sua aceitação mais difícil, já que padroniza a implementação de um número maior de conceitos. Da mesma forma, a inclusão de um nível de abstração no sistema aumenta a sua complexidade, já que aumenta a sua capacidade funcional.

Uma diferença mais relacionada com o aspecto formal dos dois documentos é que a especificação funcional do GKS é mais rica e melhor estruturada, além de apresentar padrão de "binding" com linguagem de programação (na versão ANSI).

Outras diferenças relacionadas a seguir estão mais no nível das definições das funções do que propriamente das capacidades funcionais dos sistemas.

Primitivas de Saída

- O GKS não utiliza o conceito de CP ("current position"); em consequência, as posições iniciais são passadas como parâmetros das primitivas de saída e não existem comandos relativos ou do tipo MOVE.
- As primitivas POLYLINE e POLYMARKER absorveram as primitivas LINE e MARKER, no GKS.
- O mecanismo de escape no CORE é uma função extra, enquanto

no GKS é uma primitiva de saída GDP ("Generalized Drawing Primitive").

Segmentação

- O CORE faz uma distinção entre segmentos retidos e temporários enquanto o GKS só tem segmentos retidos, permitindo, porém, que primitivas de saída sejam criadas fora de segmentos.
- No GKS, o conceito de armazenamento de segmentos independente de "workstation" permite operações com segmentos entre "workstations".

Atributos

- No GKS, o programador de aplicação pode criar representações de atributos sob forma de tabelas "bundle" para cada "workstation" e estabelecer os valores dos atributos em grupo através de um índice ou individualmente; no CORE, a capacidade de especificar atributos em grupo é limitada a tabelas pré-definidas.
- O CORE fornece um atributo estático para fixar o tipo de transformação de imagem válida para os segmentos retidos; além disso, fornece atributos que afetam todos os segmentos retidos ou que afetam determinados segmentos retidos. No GKS, todos os atributos de segmentos são dinâmicos e especificados para cada segmento.

Operações de Visualização e de Transformação

- O CORE permite "windows" retangulares inclinadas enquanto no GKS só são permitidas "windows" com lados paralelos aos

eixos coordenados.

- O GKS atribui a cada "window" e a cada "viewport" o número da transformação de normalização que lhes é associado, de forma que a transformação pode ser selecionada por esse número no decorrer do programa; no CORE, a "window" e a "viewport" correntes são sempre as últimas especificadas.
- O GKS possui uma lista de prioridades de entrada para "viewport"; tem funções para estabelecer "window" e "viewport" na transformação de "workstation" e permite composição de transformações de imagem.

Primitivas de Entrada

- O CORE considera dois subconjuntos disjuntos de dispositivos lógicos de entrada : aqueles que só causam evento e aqueles que só podem ser amostrados, permitindo associações entre esses dispositivos. No GKS, qualquer dos dispositivos lógicos pode ser operado de três modos: REQUEST, SAMPLE e EVENT, não havendo necessidade de associação.

"Interface" com o "Metafile"

- Só o GKS provê funções padronizadas para realizar a "interface" do programa de aplicação com o "metafile".

7. CONCLUSÃO

O estudo dos dois sistemas gráficos mostra que a sua origem, sua filosofia e seus propósitos são comuns: tudo começou com a consciência emanada do "espírito de Seillac", visando uma padronização que evitasse o desperdício de esforços isolados na busca de um mesmo objetivo, a portabilidade de programas de aplicações gráficas.

Fica claro que o CORE, tendo sido a primeira proposta publicada e colocada em uso, foi a base sobre a qual se desenvolveu grande parte da Computação Gráfica a partir de 1977, inclusive o próprio GKS.

O grupo que desenvolveu o GKS, certamente por estar ligado a órgãos tradicionalmente produtores de padrões, conduziu seus esforços de desenvolvimento sempre voltado ao processo de padronização, daí a sua adoção como padrão internacional pela ISO.

Na recente polêmica deflagrada com a decisão da ACM de adotar o CORE como um "padrão ACM" (setembro de 1983), ficou evidenciado que a objeção da ISO não é baseada primordialmente no conteúdo técnico do CORE, mas antes no fato de que ele não foi submetido a esse processo de padronização.

A importância da adoção de um padrão para subrotinas gráficas é inquestionável, mas a sua consolidação é uma tarefa bastante complexa. Não basta a aceitação de um certo padrão pelos órgãos de padronização para que ele se torne um padrão de fato,

já que essas decisões tem implicações político-econômicas; além disso, não se pode desconsiderar a força dos "softwares" básicos distribuídos por fabricantes de dispositivos.

Talvez o mais importante em toda essa discussão seja a filosofia de programação e utilização de sistemas gráficos. Essa filosofia é uniforme nos padrões propostos e a sua adoção acarretará no maior entendimento da área e no seu consequente desenvolvimento.

BIBLIOGRAFIA

(1) Status Report of the Graphics Standards Planning Committee, ACM SIGGRAPH, Computer Graphics, 13(3), Agosto 1979.

(2) ISO; Information Processing - Graphical Kernel System (GKS) - Functional Description, ISO DIS 7942, Novembro 1982.

(3) ANSI/X3H3; Graphical Kernel System, Computer Graphics, Special GKS Issue, Fevereiro 1984.

(4) W. Newman e A. van Dam; Recent Efforts Towards Graphics Standardization, Comp. Surv. 10(4), Dezembro 1978.

(5) J.D. Foley e A. van Dam; Fundamentals of Interactive Computer Graphics, Addison-Wesley, Washington, 1983.

(6) J. Encarnação, G. Enderle e outros; The Workstation Concept of GKS and the Resulting Conceptual Differences to the GSPC CORE System, Computer Graphics, 14(3), Julho 1980.

(7) F.E. Langhorst e T.B. Clarkson III; Realizing Graphics Standards for Microcomputers, Byte, Fevereiro 1983.

(8) C. Card, R.D. Prigge, J.L. Walkowicz e M.F. Hill; The World of Standards, Byte, Fevereiro 1983.

(9) C.J. Date; An Introduction to Database Systems, Addison-Wesley, USA, 1981.

(10) Codasyl Data Base Task Group Report, Abril 1971.

(11) D.A. Menascé e D. Schwabe; Redes de Computadores - Aspectos Técnicos e Operacionais, Terceira Escola de Computação, DI-PUC/RJ, 1982.

(12) T.W. Pratt; Programming Languages Design and Implementation, Prentice-Hall, USA, 1975.

(13) E.I. Organick, A.I. Forsythe, R.P. Plummer; Programming Language Structures, Academic Press, USA, 1978.

(14) P.J.W. Hazen; Activities of ISO/TC97/SC5/WG2 and their Impact in Europe (Abstract) , Computer Graphics 16(3), Julho 1982.

(15) L. Hatfield; GKS and the Alphabet Soup of Graphics Standards (An Informal Comentary), Computer Graphics 16(2), Junho 1982.

(16) J.C. Michener e A. van Dam; A Functional Overview of the CORE System with Glossary; Comput. Surv. 10(4), Dezembro 1978.

(17) G. Enderle, K. Kansy e G. Pfaff; Computer Graphics Programming - GKS - The Graphics Standard, Springer-Verlag, Berlin, 1984.

(18) E. L. Sonderegger; Comparison of Proposed 3D Graphics Standards, Computer Graphics 17(4), Outubro 1983.

(19) E. L. Sonderegger; Report on Current Graphics Standards Activities, Computer Graphics 18(1), Janeiro 1984.

(20) ACM-SIGGRAPH : Letters on CORE System Standardization, Computer Graphics 18(1), Janeiro 1984.