UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELETRICA

DEPARTAMENTO DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

GAV

UM GERENCIADOR DE AREAS DE VISUALIZAÇÃO

BASEADO NA NORMA GKS

Este exemplar corresponde a redação final da tese de mestrado defendida por Helga I.M. Hartelt, aprovada pela comissão julgadora em 10/12/86.

a

por: Eng. Helga Ingrid Mendes Hartelt

Orientador: Prof. Dr. Léo Pini Magalhães

Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas

DEZEMBRO DE 1986

Este trabalho contou com o apoio financeiro dos seguintes órgãos:

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

- CAPES -

Fundação de Amparo à Pesquisa do Estado de São Paulo

- FAPESP -

dedico à

minha família

AGRADECIMENTOS

Ao professor Léo Pini Magalhães, pela confiança, incentivo e orientação.

Ao professor Valdemar Cardoso da Rocha Junior, pela amizade e apoio durante a graduação, que me incentivaram a realizar esse curso de mestrado.

Aos professores Beatriz Mascia Daltrini e Clésio Luis Tozzi, pelas sugestões e discussões durante a elaboração do trabalho.

À colega Maria Nuria Pujol Andrés, pela valiosa colaboração na implementação das interfaces GKS-DISPOSITIVO (normal e especial) para o terminal VT240.

Aos funcionários do Centro de Computação da UNICAMP responsáveis pelo sistema VAX-UNICAMP, pelo apoio e amizade dispensados durante a fase final de implementação deste trabalho.

Ao CTI, pela utilização do sistema computacional PCS-CADMUS.

À CAPES e à FAPESP, pelo apoio financeiro.

Aos colegas Armando, Edvaldo, Heraldo, Ivan e Sérgio, pela convivência agradável nesse período.

A Rubens, Ivonete e Ting, pelo apoio nos momentos difíceis.

E a todos aqueles que colaboraram, direta ou indiretamente, para a realização deste trabalho.

RESUMO

Neste trabalho, um Gerenciador de Recursos Gráficos para aplicação em Sistemas de Gerenciamento de Interfaces de Usuário (SGIU), o Gerenciador de Áreas de Visualização (GAV), é apresentado considerando-se o uso de um pacote gráfico GKS como suporte a esse sistema.

O GAV permite que várias áreas de visualização sejam definidas numa mesma tela útil, pelos projetistas dos programas de aplicação. Cada uma dessas áreas de visualização é vista pelo GKS como uma estação de trabalho.

O GAV permite ainda, que partes distintas do programa de aplicação sejam realizadas por diferentes usuários, podendo esses mesmos usuários, utilizar em cada uma das partes, todos os recursos oferecidos pelo GKS, sem precisar se preocupar com a existência das demais partes.

Finalmente, o GAV permite o agrupamento de estações de trabalho em conjuntos chamados configurações de tela.

ABSTRACT

In this work, a Graphics Resources Management System to be used in User Interface Management Systems, the GAV, is presented, considering the use of a GKS graphic system as a supporting tool.

The GAV allows the definition of several areas of visualization in the same screen by the designers of the application programs. Each of these areas is a workstation according to the concept of workstation established in GKS.

The GAV even allows that different parts of the application programs be done by different users. These users can use in each part, the full set of GKS functions, without worry about the existence of the other parts.

Finally, the GAV allows the grouping of workstations in sets called screen configurations.

INDICE

1-INTRODUÇÃO

Nos últimos anos, o avanço da microeletrônica e a diminuição do custo do hardware permitiram a popularização dos sistemas interativos /ENC-79/.

Com essa popularização, surgiu a necessidade desses sistemas suportarem uma grande variedade de usuários (com diferentes habilidades, interesses e níveis de conhecimento), de aplicações e de dispositivos físicos de entrada e saída, como mesas digitalizadoras, "plotters", "mouses", etc /PFA-83/.

Várias pesquisas foram realizadas nesta área e como resultado verificou-se que a interface de usuário constituía um fator determinante do sucesso ou fracasso de um sistema interativo. Constatou-se pois, a necessidade de se projetar e implementar interfaces de usuário fáceis de entender e de operar.

No entanto, o projeto e a implementação de interfaces de usuário amigáveis não é uma tarefa fácil, pois exige uma ação interdisciplinar envolvendo cientistas de computação, psicólogos industriais e cognitivos, projetistas gráficos, artistas e usuários /GIIT-83/.

Além disso, o projeto de interfaces de usuário amigáveis envolve incertezas relacionadas com /HAN-84/:

- conhecimento

  (com que frequência o operador utiliza um computador como ferramenta de trabalho)

- habilidade

  (com que frequência o operador utiliza o programa de aplicação e de que maneira ele foi treinado para utilizá-lo)

- preferências pessoais

  (que métodos de comunicação o operador prefere)

- "physiognomy"

  (que tipos de dispositivos físicos são preferidos por uma comunidade de usuários).

Assim, diferentes usuários, com seus diferentes níveis de conhecimento, habilidades e preferências pessoais, exigem características funcionais distintas para a interface de usuário. Um usuário inexperiente, por exemplo, deseja mais informações a cada passo do diálogo com o sistema do que aquelas exigidas por um usuário experiente.

Esses fatos levaram à idéia de projeto e implementação de interfaces de usuário ajustáveis às características do usuário.

Por outro lado, verificou-se que nenhum diálogo, tendo-se em vista a sua utilização em uma aplicação específica, pode ser implementado a priori, totalmente otimizado. É necessário adaptá-lo à aplicação a que ele se destina.

Surge assim, a necessidade de se criar ferramentas que possibilitem a avaliação do desempenho de um diálogo e a sua adaptação segundo resultados obtidos nessa avaliação e daí o conceito de Sistemas de Gerenciamento de Interfaces de Usuário (SGIU).

Um SGIU é um conjunto de ferramentas de software para apoio a construção e controle de diálogos /THO-85/.

As ferramentas que constituem um SGIU devem apoiar o trabalho dos projetistas de diálogo durante as três fases de

projeto de um diálogo: definição, "prototyping" e geração /SIL-85/.

O presente trabalho visa dar continuação ao trabalho de definição de um SGIU, o Processador de Diálogo, realizado em /SIL-85/.

Ele consiste na definição e implementação de um dos componentes do Processador de Diálogos, o Gerenciador de Recursos Gráficos, referido a partir de agora como Gerenciador de Áreas de Visualização (GAV).

O GAV utiliza como suporte um pacote gráfico GKS. Isso porque a norma GKS é a norma internacional, estabelecida pela ISO, para projeto e implementação de pacotes gráficos. A relação custo/benefício desta solução será enfocada em um dos capítulos do trabalho.

No capítulo 2 os SGIU são abordados em detalhe. Nesse capítulo esses sistemas são definidos e são apontadas algumas das vantagens da sua utilização em sistemas interativos.

O capítulo 3 apresenta o SGIU proposto em /SIL-85/, o Processador de Diálogo, sendo o GAV situado em seu contexto.

O capítulo 4 apresenta o Gerenciador de Áreas de Visualização, abordando seus conceitos básicos, a visão por ele oferecida aos usuários e sua definição funcional.

No capítulo 5 é fornecido o projeto detalhado do GAV.

A presença do GAV em um sistema gráfico interativo exige uma interface GKS-DISPOSITIVO GRAFICO com características especiais. Essa interface é o assunto abordado no capítulo 6.

O capítulo 7 analisa o uso de um pacote gráfico GKS como suporte ao GAV, apontando as vantagens e as desvantagens

desta solução.

O capítulo 8 apresenta um exemplo de utilização do GAV.

O capítulo 9 contém as considerações finais.

O apêndice A fornece uma descrição funcional do GAV, de uma forma semelhante àquela utilizada pela norma GKS, para descrever as suas funções.

O apêndice B fornece o resultado da implementação do exemplo de utilização do GAV.

2- SISTEMAS DE GERENCIAMENTO DE INTERFACE DE USUARIO

2.1- INTRODUÇÃO

A fácil aceitação de um sistema pela comunidade de usuários está intimamente relacionada com a qualidade da interface homem-máquina oferecida por esse sistema. Uma interface homem-máquina mal projetada pode comprometer o sucesso de um sistema inteiro.

A partir do reconhecimento deste fato, maior importância passou a ser dada ao projeto e implementação de interfaces de usuário. Novos métodos e técnicas surgiram e foram postos em discussão.

Dessa maior atenção reservada ao problema, verificou-se que a produção de interfaces de usuário amigáveis exigia uma ação interdisciplinar, envolvendo cientistas de computação, psicólogos industriais e cognitivos, projetistas gráficos, artistas e usuários /GIIT-83/.

Verificou-se ainda, que a definição de uma camada de diálogo, responsável pelo controle da interação e distinta da camada de aplicação, responsável pela execução dos algoritmos referentes à aplicação propriamente dita, traz diversas vantagens. Entre as quais /SIL-85/:

- permite a exploração de várias características de dispositivos físicos e de técnicas de interação para atender aos fatores humanos do operador;
- traz economia de custos, facilitando a construção da interface de usuário, permitindo seu aprimoramento para casos específicos

e a sua reutilização por outras aplicações;

- permite manter a consistência através de várias aplicações que podem fazer uso da mesma interface.

Dessa maneira, o programador de aplicação pode projetar suas rotinas considerando que as rotinas de interação, necessárias para realizar a troca de informações com o usuário, foram previamente implementadas e encontram-se ao seu dispor.

Certamente, a maior vantagem oferecida por esse esquema, é a possibilidade de aproveitamento do esforço empregado no projeto e implementação de um diálogo para uma aplicação específica, no projeto e implementação de um outro diálogo a ser utilizado, possivelmente, por outra aplicação.

No entanto, é importante salientar que nenhum diálogo, tendo-se em vista a sua utilização em uma aplicação específica, pode ser implementado, a priori, totalmente otimizado, ou seja, esse aproveitamento de esforço não é total.

Surge assim, a necessidade de se criar ferramentas que possibilitem a avaliação do desempenho de um diálogo e a sua adaptação segundo os resultados obtidos nessa avaliação. Surge então, o conceito de Sistemas de Gerenciamento de Interfaces de Usuário (SGIU).

Um SGIU é um conjunto de ferramentas de software para construção e controle de diálogos /THO-85/. A sua utilização em um sistema interativo facilita o desenvolvimento de programas de aplicação no tocante ao projeto e a implementação de interfaces de usuário mais consistentes e mais fáceis de serem mantidas.

Nesse capítulo, os SGIU serão definidos e especificados

funcionalmente, as vantagens de sua utilização durante a fase de projeto e implementação de um diálogo serão apontadas, será apresentado o modelo de um sistema interativo com SGIU e serão especificadas as ferramentas que o compõem.

2.2- HISTORICO

As necessidades e os conceitos que levaram à definição de SGIU foram discutidos durante muitos anos (BUXTON e SNIDERMAN 1980; FOLEY 1981; SWARTOUT e BOLGER 1982).

Após alguns esforços concentrados no tema surgiram os primeiros exemplos de SGIU. O SEEDIS, no Lawrence Berkely Laboratories e o ALDS (Thomas e Hall 1983) /THO-85/ podem ser citados.

A partir daí, vários pesquisadores interessaram-se pelo assunto, e passaram a preocupar-se com a formalização de ferramentas para projeto de interfaces de usuário.

Para ajudar a estabelecer os conceitos relacionados com SGIU, foi proposto um encontro sobre "Graphic Input Interaction Techniques" que aconteceu em Seatle. Nesse encontro, vários conceitos importantes sobre SGIU foram firmados e a necessidade de cooperação e estudo interdisciplinar foi fortificada. Desde então, diversas pesquisas foram realizadas nas diversas áreas que abrange um SGIU /THO-85/.

Atualmente, vários SGIU foram implementados e alguns estão sendo comercializados. Entre eles encontram-se: SYNGRAPH, TIGER/UIMS, DMS, REXX/FSX /OLSEN-84/.

2.3- EQUIPE NECESSARIA PARA O DESENVOLVIMENTO DE SISTEMAS INTERATIVOS

Antes de prosseguir falando sobre os SGIU é importante identificar os papéis a serem exercidos por uma equipe, empenhada em um projeto de desenvolvimento de um sistema interativo. Estes papéis estão claramente definidos em /OLSEN-84/ e são os seguintes:

1- USUARIO FINAL- sua função é utilizar o sistema interativo implementado. Sob seu ponto de vista, o sistema deve ser o mais amigável possível.

2- ANALISTA DA APLICAÇÃO- provê uma descrição funcional da aplicação que está sendo implementada. E um especialista na área da aplicação. Juntamente com o usuário final, é responsável pela definição e especificação do problema que o sistema interativo pretende resolver.

3- PROGRAMADOR DA APLICAÇÃO- produz os módulos referentes à aplicação propriamente dita. Deve preocupar-se em produzir programas eficientes, reusáveis e de fácil manutenção.

4- PROJETISTA DE DIALOGO- deve projetar e implementar os módulos de diálogo, responsáveis pela comunicação homem-máquina. O autor de diálogos deve preocupar-se em prover formas de interação naturais e diálogos consistentes e facilmente compreensíveis. Ele deve estar familiarizado com a aplicação e com vários estilos de interação e dispositivos e ainda, deve conhecer as características do usuário final.

5- PROJETISTA GRAFICO- determina as diversas configurações de tela, ou seja, determina que informações deverão aparecer na

tela a cada instante.

6- AVALIADOR DE DIALOGO- analisa a performance do diálogo.

7- IMPLEMENTADOR DO SGIU- deve projetar e implementar ferramentas que auxiliem o trabalho do autor de diálogos, do avaliador de diálogos e do projetista gráfico. Deve ainda, fornecer um ambiente que suporte a execução de sistemas interativos, tendo, para isso, que possuir conhecimento em linguagens de programação, sistemas operacionais, algoritmos gráficos, etc.

8- PROJETISTA DE AMBIENTE- responsável pelo projeto do ambiente físico, onde será instalado o sistema interativo.

9- AVALIADOR DE AMBIENTES- responsável pela avaliação do ambiente físico.

2.4- DEFINIÇÃO

Um SGIU é um conjunto de ferramentas de software para apoio à construção e controle de diálogos /THO-85/.

Estas ferramentas podem ser classificadas em quatro grupos /OLSEN-84/:

-Ferramentas de implementação

As ferramentas de implementação são aquelas já encontradas na maioria dos sistemas computacionais. Os editores, as linguagens de programação e as ferramentas de gerenciamento de software constituem esse grupo.

-Ferramentas de suporte interno

Essas ferramentas suportam a interface de usuário internamente e não se encontram necessariamente ao alcance do

autor de diálogo ou do projetista gráfico. São exemplos de ferramentas neste grupo: pacotes gráficos, gerenciadores de janelas, analisadores gramaticais, etc.

- Ferramentas de especificação

Esse grupo de ferramentas dá apoio ao trabalho do autor de diálogo e do projetista gráfico. Elas permitem que a descrição dos protótipos de diálogo seja realizada de uma maneira mais fácil.

As ferramentas de especificação devem suportar e encorajar o projeto de interfaces de usuário amigáveis.

-Ferramentas de avaliação

As ferramentas de avaliação auxiliam o trabalho do avaliador de diálogo, podendo atuar tanto durante a fase de especificação dos diversos protótipos de diálogo, como durante a sua execução.

As ferramentas de avaliação utilizadas durante a especificação do diálogo desempenham um papel importante na prevenção da implementação de interfaces pouco amigáveis. Já aquelas que atuam durante a fase de execução dos diversos protótipos de diálogo, encarregam-se de obter informações a respeito do desempenho do usuário final e de analisá-las.

2.5- FUNÇOES DESEMPENHADAS POR UM SGIU

Um SGIU deve servir de mediador entre o programa de aplicação e o usuário final. Assim, quando o programa de aplicação necessita de um dado, ele recorre ao SGIU, que aciona o módulo de diálogo responsável por aquela interação.

Além disso, um SGIU deve oferecer ferramentas para auxílio do projetista de diálogo, do avaliador de diálogo e do projetista gráfico, durante a fase de projeto e implementação de um diálogo. A partir do uso dessas ferramentas, esses especialistas podem:

1- especificar vários protótipos de diálogo;

2- executar cada um desses protótipos, juntamente com a aplicação, em diversas condições, colhendo dados sobre seu comportamento;

3- avaliar os dados coletados, determinando qual o protótipo de diálogo mais adequado para aquela aplicação específica.

Portanto, um SGIU atua tanto durante a fase de projeto e implementação de um diálogo, como durante a sua execução normal.

Assim, resumidamente, pode-se dizer que um SGIU deve ser capaz de:

- gerenciar a comunicação entre o usuário final e o sistema computacional;
- oferecer recursos ao projetista de diálogo, ao avaliador de diálogo e ao projetista gráfico que possibilitem a definição e o "prototyping" de diálogos, permitindo assim a obtenção de diálogos ótimos, segundo uma avaliação, para uma aplicação específica.

2.6- VANTAGENS DA UTILIZAÇÃO DE SGIU

São diversas as vantagens advindas do uso de um SGIU durante o projeto e a implementação de diálogos. Entre elas,

pode-se citar /OLSEN-84/:

- desenvolvimento de interfaces de usuário mais consistentes;
- especificações dos módulos de diálogo facilmente representadas, validadas e avaliadas;
- projetos de diálogos podem ser rapidamente submetidos a "prototyping" e implementados;
- manutenção de todo o sistema interativo torna-se mais rápida e barata.

2.7- MODELO DE SISTEMA INTERATIVO COM SGIU

O modelo de um sistema gráfico interativo com um SGIU pode ser visto na figura 2.1 /MAG-85/.

Esse modelo ressalta o fato de que toda a entrada e saída de dados é controlada pelo SGIU. Considerando o modelo apresentado na figura 2.1, a especificação, o projeto e a implementação do diálogo entre o usuário e o sistema podem ser realizados de maneira independente do projeto e implementação da aplicação propriamente dita.

2.8- IMPACTOS DA EVOLUÇÃO DOS SGIU NA COMUNIDADE DE USUARIOS

A evolução dos SGIU, nos últimos anos, produziu impactos na comunidade de usuários desses sistemas.

Segundo /OLSEN-84/, a maior implicação da evolução dos SGIU, para o usuário final, foi a redução do custo de desenvolvimento de novas aplicações interativas, decorrente de um maior aproveitamento do esforço dispendido no projeto de um

diálogo para uma aplicação específica, no projeto de um diálogo a ser utilizado por outra aplicação.

O programador de aplicação é liberado dos problemas relacionados com o projeto e a implementação do diálogo e se concentra apenas nos problemas relacionados com a solução algorítmica.

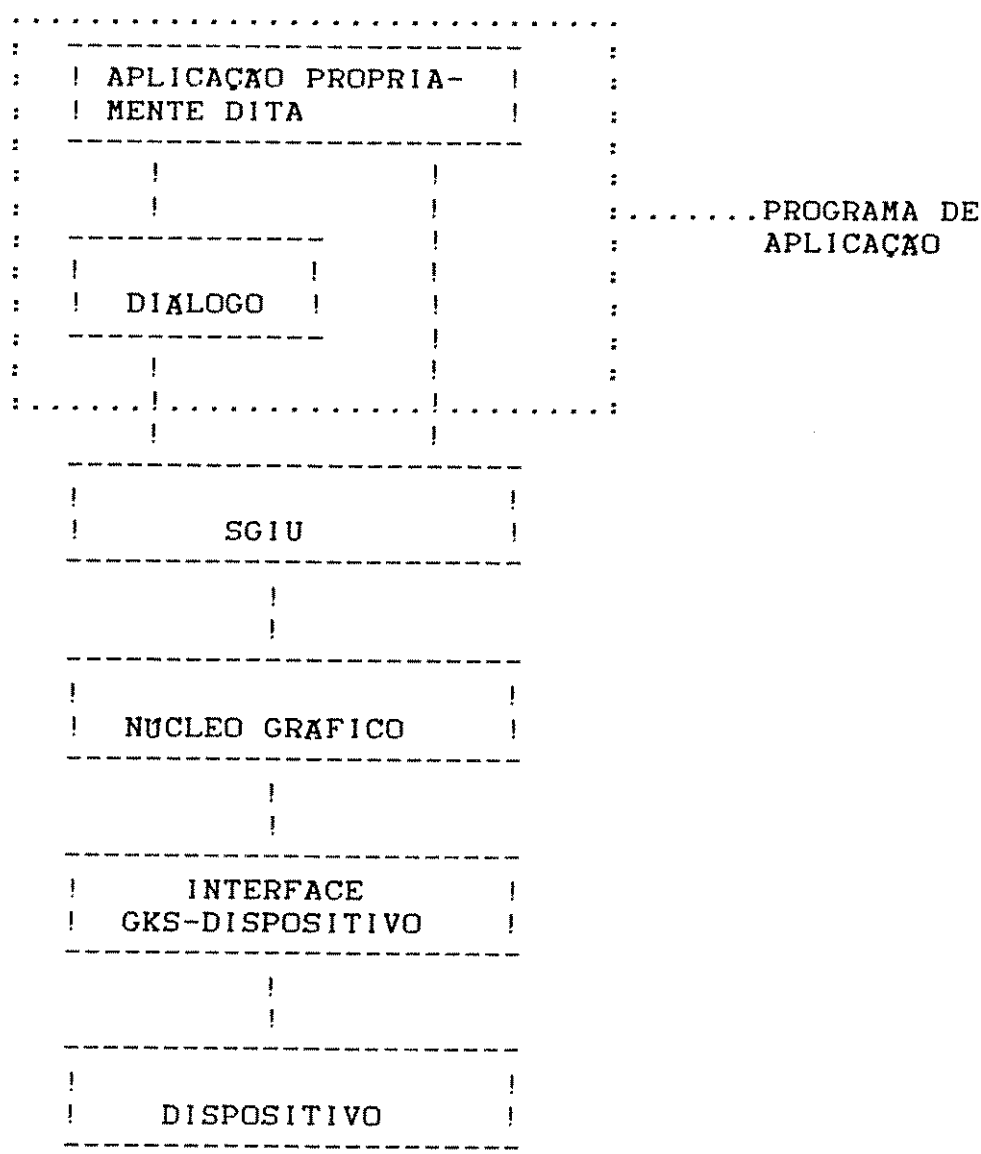

fig.2.1 Modelo de um sistema interativo com SGIU

O autor de diálogos, por sua vez, pode concentrar-se no projeto e implementação do diálogo. O diálogo apropriado é determinado com a aplicação do "loop": projeto/ "prototyping"/ avaliação, com a participação intensa do usuário final.

O projetista gráfico e o avaliador de diálogos tiveram seu trabalho facilitado pelas diversas ferramentas postas a sua disposição pelos SGIU's e em alguns casos, a redução do custo do projeto e implementação do diálogo tornou economicamente viável a produção de diálogos altamente adaptados à aplicação.

No capítulo seguinte, um SGIU definido em /SIL-85/, o Processador de Diálogo, é apresentado. O GAV é então, situado em seu contexto.

3- PROCESSADOR DE DIALOGO: PROPOSTA DE UM SGIU

3.1- INTRODUÇAO

Este capítulo focaliza um conjunto de ferramentas para geração de diálogos, o Processador de Diálogo, definido em /SIL-85/.

A área de Aplicação de Metodologia de Projeto de Software pode ser citada como sendo a área de aplicação do Processador de Diálogo, devido a sua capacidade de apoio ao projeto de diálogos.

O Processador de Diálogo implementa um SGIU e como tal deve auxiliar o projetista na produção de diálogos, permitindo a definição, o "prototyping" e a geração de um diálogo otimizado.

Esse sistema permite selecionar a melhor estratégia para operadores específicos operarem programas também específicos, permitindo assim, a otimização da comunicação operador-sistema.

O Processador de Diálogo pode assim, ser visto como um sistema gerador de protótipos. Ele oferece várias facilidades ao projetista de diálogo, para a avaliação de diversas possibilidades para a estrutura de um diálogo.

Uma vez identificada a estrutura de diálogo ótima para o diálogo entre um operador e uma aplicação específicos, o Processador de Diálogo permite a geração de um produto final, constituído pela aplicação propriamente dita mais o diálogo ótimo, de onde todos os componentes do processador, utilizados no "prototyping" e não mais necessários à execução da aplicação interativa, foram retirados.

A seguir será apresentada a definição funcional do Processador de Diálogo e logo após, será descrita a topologia deste sistema.

Este capítulo tem como objetivo situar o GAV no contexto do Processador de Diálogo.

3.2- DEFINIÇÃO FUNCIONAL DO PROCESSADOR DE DIALOGO

O Processador de Diálogo é um conjunto de ferramentas de software, colocadas à disposição dos projetistas de diálogo para a criação de diálogos.

A seguir, serão especificadas as funções do Processador de Diálogo, durante as três fases do projeto e implementação de um diálogo.

3.2.1- DEFINIÇÃO DE DIALOGOS

Nesta fase os protótipos de diálogo são definidos. Cada protótipo representa uma possível estrutura de diálogo para a aplicação em questão.

São as seguintes, as funções do Processador de Diálogo, na fase de definição:

- modelamento do diálogo;
- pré-processamento;
- análise da definição.

O Processador de Diálogo deve oferecer uma linguagem, a Linguagem de Especificação de Diálogos (LED), que permite a

descrição dos protótipos.

Essa linguagem deve conter primitivas de alto nível que permitam uma fácil descrição de todos os passos de um diálogo.

A LED pode ser definida, como por exemplo em /SIL-85/, como uma extensão de uma linguagem de alto nível já existente. E necessário, pois, um pré-processamento e posteriormente uma compilação, para a obtenção do diálogo em código objeto.

3.2.2- "PROTOTYPING"

O termo "prototyping", de uma forma genérica, se aplica ao conjunto de tarefas que envolvem a execução, a avaliação e a adaptação de protótipos.

E função do Processador de Diálogo, oferecer recursos que facilitem o "prototyping" de diálogos. Esses recursos devem permitir a execução de diversos protótipos de diálogo, a coleta de dados referentes à execução de cada um desses protótipos, a análise dos dados coletados, a avaliação dos resultados da análise e a adaptação dos protótipos segundo os resultados dessa avaliação.

A adaptação do protótipo de diálogo pode ser realizada /SIL-85/:

- a partir de alterações na estrutura do diálogo que o tornem mais simples ou mais inteligente (adaptação off-line);
- através da escolha de um outro protótipo, contido na biblioteca de diálogos, que se adeque melhor ao usuário (adaptação on-line).

E produto desta fase, portanto, o diálogo-ótimo. Esse diálogo, no entanto, contém as primitivas específicas de

"prototyping".

3.2.3- GERAÇÃO

Na última das três fases de projeto e implementação de um diálogo, a de geração, são retiradas do sistema aplicação propriamente dita + diálogo-ótimo, todas as primitivas da LED específicas de "prototyping", ou seja, cuja atuação se resume à fase de "prototyping". Essa é a função do Processador de Diálogo nesta fase.

3.3- TOPOLOGIA DO PROCESSADOR DE DIALOGO

O objetivo deste item é descrever os componentes do Processador de Diálogo e suas interligações, tendo-se em mente as funções realizadas por esse sistema, durante cada uma das três fases do projeto e implementação de um diálogo e durante a execução normal de um diálogo.

Alguns dos componentes do Processador de Diálogo realizam funções off-line em relação à execução do protótipo de diálogo (por exemplo: o pré-processador), já outros realizam funções on-line (por exemplo: o coletor de dados).

3.3.1- DEFINIÇÃO

Os componentes necessários nesta fase são: editor, pré-processador, compilador e ligador.

O editor permite a edição dos diversos subprogramas-diálogo. Um subprograma-diálogo é definido como sendo cada uma das partes de um diálogo, capaz de realizar uma interação com o operador. Assim pode-se ter diversos subprogramas-diálogo capazes de atender a um pedido de dado realizado pela aplicação. A otimização do diálogo está em se determinar, para cada interação, qual é o subprograma-diálogo mais adequado.

O pré-processador lê cada um dos subprogramas-diálogo, interpreta-os e gera rotinas na linguagem de alto nível, que serviu de base para a LED. Faz parte da função do pré-processador a análise do código que permita a detecção de erros.

O ligador permite a obtenção da tarefa-aplicação que constitui-se de:

1- programa de aplicação;

2- supervisor de execução;

3- subprogramas-diálogo;

4- utilitários de "prototyping";

5- rotinas do núcleo gráfico.

A sequência de atividades, realizadas durante a fase de definição, pode ser vista na fig.3.1.

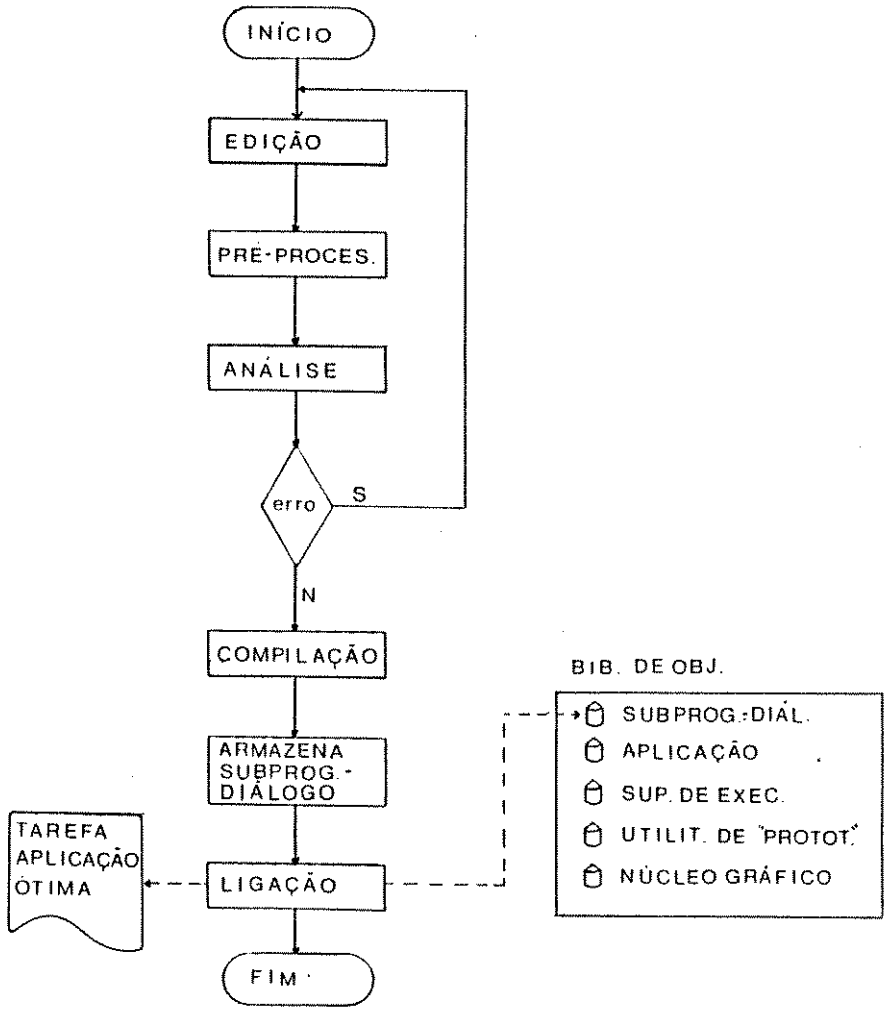

fig. 3.1- Atividades realizadas durante a definição

3.3.2- "PROTOTYPING"

Nesta fase a tarefa-aplicação, obtida na definição, é executada. Os componentes atuantes são, o supervisor de execução e os utilitários de "prototyping".

Quando a aplicação deseja comunicar-se com o usuário, ela aciona o Supervisor de Execução, que por sua vez aciona um dos subprogramas-diálogo providos para aquela interação e passa-lhe o controle de execução. O subprograma-diálogo é então, executado e dados são coletados e possivelmente analisados, através do acionamento dos utilitários de "prototyping". O controle de execução é, então, devolvido ao Supervisor de Execução, que aciona outro subprograma-diálogo, definido para a mesma interação, repetindo-se o processo. Quando não há mais subprogramas-diálogo previstos para aquela interação, o controle de execução é devolvido à aplicação.

Assim, a cada nova solicitação de interação, realizada pela aplicação, um grupo de subprogramas-diálogo é executado e avaliado, possibilitando a determinação do subprograma-diálogo ótimo.

Ao final da execução da tarefa-aplicação, pode-se determinar pelos dados coletados, qual é o subprograma-diálogo apropriado para cada interação e o conjunto desses subprogramas-diálogo constitui o diálogo otimizado para aquela aplicação específica.

Pode-se assim, perceber a necessidade de três elementos fundamentais no Supervisor de Execução:

- Selecionador de Sequências de Diálogo;

- Supervisor de "Prototyping"
- Gerenciador de Recursos Gráficos.

O Selecionador de Seqüências é responsável por, mediante a solicitação da aplicação, determinar quais são (no "prototyping") ou qual é (na execução normal diálogo) o(s) subprograma(s)-diálogo previsto(s) para aquela interação específica.

O Supervisor de "Prototyping" garante a execução de todos os subprogramas-diálogo, providos para atender aquela interação, para cada solicitação de interação realizada pela aplicação. Observa-se, então, que o Supervisor de "Prototyping" atua somente durante a fase de "prototyping", não sendo necessária a sua presença durante a execução normal do diálogo.

O Gerenciador de Recursos Gráficos é o módulo responsável pelo gerenciamento da utilização dos recursos gráficos pela aplicação e pelo diálogo, conforme figura 3.2. Esse módulo será abordado com detalhes nos capítulos que se seguem, sendo referido como Gerenciador de Áreas de Visualização (GAV).

A figura 3.2 apresenta o diagrama da tarefa-aplicação sendo executada. Neste exemplo, o programa de aplicação solicita m interações com o usuário e são providos n subprogramas-diálogo para cada uma dessas interações.

3.3.3- GERAÇÃO

Na fase de geração, atuam os seguintes componentes: o limpador, o pré-processador, o analisador de definições, o

compilador e o ligador.

O limpador incumbe-se de retirar do(s) subprograma(s)-diálogo, selecionados na fase anterior, todas as primitivas específicas da fase de "prototyping" e desnecessárias à operação normal do diálogo.

Os demais componentes já tiveram suas funções descritas anteriormente.

A figura 3.3 descreve a obtenção da tarefa-aplicação ótima. E importante observar nesta figura, a utilização de uma versão simplificada do Supervisor de Execução. Essa versão simplificada nada mais é, senão o Supervisor de Execução desprovido de um de seus componentes, o Supervisor de "Prototyping", uma vez que sua atuação não é mais necessária.

3.4- CONCLUSÃO

Neste capítulo, o Processador de Diálogo, uma ferramenta para o apoio ao projeto e implementação de diálogos, definido em /SIL-85/, foi apresentado.

O Gerenciador de Áreas de Visualização (GAV), tema do restante desse trabalho, foi situado como um componente do Processador de Diálogo.

Nos próximos capítulos o GAV será apresentado, através de uma especificação funcional e será fornecido seu projeto detalhado.

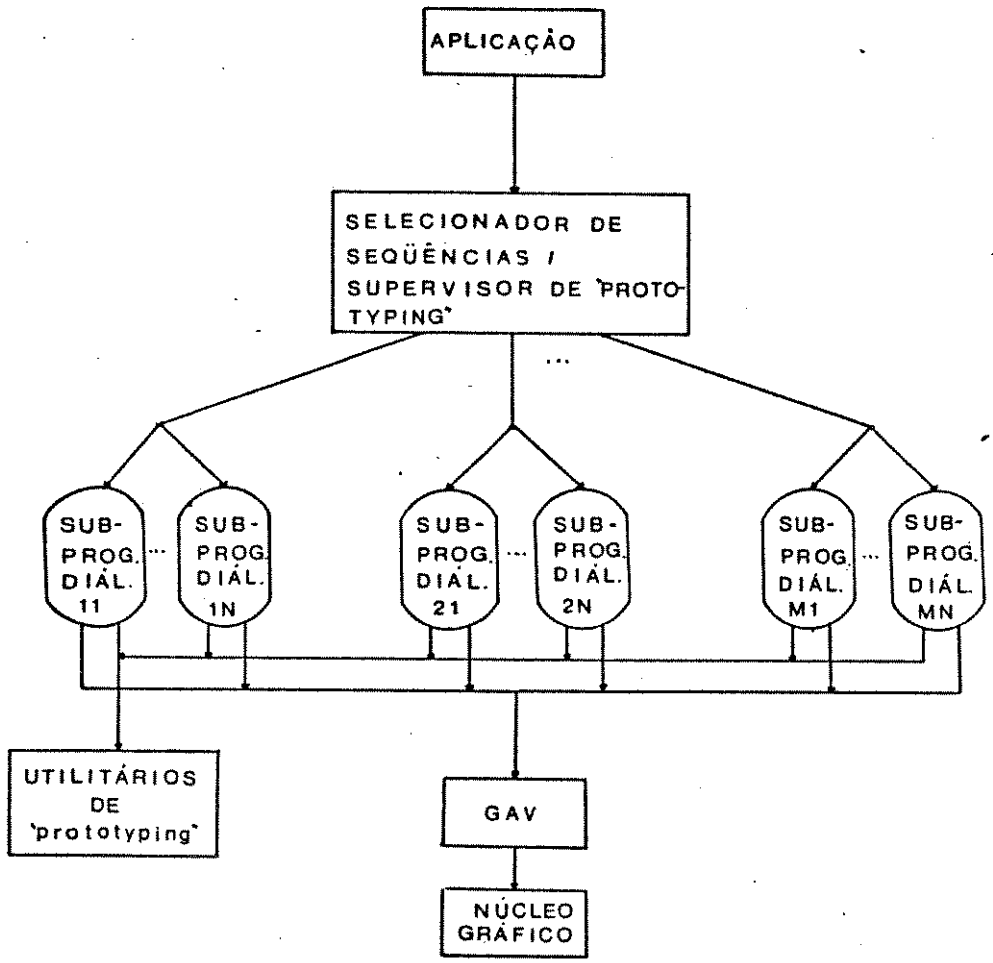

fig. 3.2 - Diagrama da tarefa-aplicação sendo executada

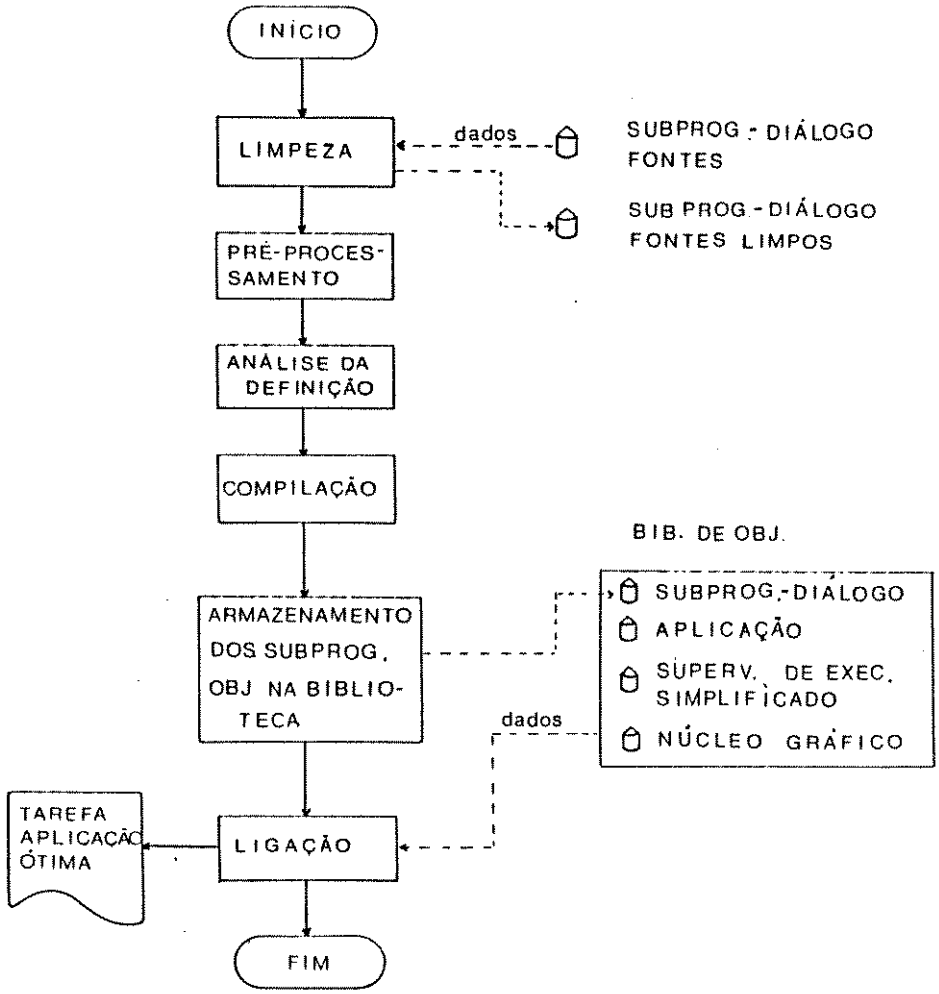

fig.3.3-Seqüência de atividades realizadas na fase de geração

4- APRESENTAÇÃO DO GERENCIADOR DE AREAS DE VISUALIZAÇÃO

4.1- INTRODUÇÃO

Neste capítulo, um Gerenciador de Recursos Gráficos para aplicação em SGIU, o Gerenciador de Áreas de Visualização (GAV) é apresentado, considerando-se o uso de um pacote gráfico, baseado na norma GKS, como suporte a esse sistema.

Esse sistema, o GAV, permite que várias áreas de visualização independentes, sejam definidas na mesma tela útil (superfície de visualização fisicamente disponível) pelos projetistas dos programas de aplicação (referidos neste trabalho, a partir de agora, como usuários). Cada uma dessas áreas de visualização é vista pelo GKS como uma estação de trabalho.

O GAV permite ainda, que partes distintas do programa de aplicação, sejam realizadas por diferentes usuários, podendo esses mesmos usuários, utilizar em cada uma das partes, todos os recursos oferecidos pelo GKS, sem precisar se preocupar com a existência das demais partes. Isso é possível devido ao conceito de processo visual, introduzido pelo GAV.

Finalmente, o GAV permite o agrupamento de estações de trabalho em conjuntos chamados configurações de tela. Os dados das estações de trabalho que compõem uma configuração de tela podem ser vistos simultaneamente na tela útil, enquanto essa configuração de tela for a corrente. Vale salientar que não há maneira de apresentar ao mesmo tempo, na tela útil, dados pertencentes a estações de trabalho que não façam parte da mesma configuração de tela.

Neste trabalho está sendo considerado o uso de um

pacote gráfico baseado na norma GKS como suporte ao GAV. Esta solução oferece como vantagem principal o projeto e a implementação de um sistema altamente portátil, uma vez que o GKS é o padrão internacional para construção de pacotes gráficos.

Pode-se observar que o GAV, na verdade extende o pacote gráfico GKS. Ele coloca certos recursos, que não são oferecidos pelo pacote gráfico, à disposição do usuário.

A seguir, o GAV é definido e tem seus conceitos básicos e sua estrutura de dados apresentados.

4.2- DEFINIÇAO DO GAV

Pode-se definir o GAV como sendo um sistema de software, cuja função é gerenciar a utilização de diversas áreas de visualização pelos vários módulos, que compõem um programa de aplicação.

Cada um desses módulos pode utilizar todos os recursos oferecidos pelo pacote gráfico e pode dispor de várias áreas de visualização independentes.

O referido sistema apóia-se em dois conceitos básicos: processo visual e configuração de tela.

4.2.1- PROCESSO VISUAL

Considerando um programa de aplicação composto por vários módulos e envolvendo diversas áreas de conhecimento (por exemplo, em uma aplicação interativa pode-se distinguir ao menos

dois módulos como elementos constituintes do programa de aplicação: um responsável pela execução dos algoritmos da aplicação propriamente dita e outro pela comunicação usuário-sistema), é desejável que esses módulos sejam projetados por especialistas nas respectivas áreas.

Tomando por hipótese o fato de que esses especialistas, utilizarão os recursos do pacote gráfico, devem ser oferecidos recursos, pelo GAV, que facilitem o trabalho desses usuários. Esses recursos devem possibilitar o uso pleno do pacote gráfico, no contexto de cada usuário, de uma forma independente dos demais.

Pode-se definir um processo visual como sendo cada parte do programa de aplicação, que utiliza um conjunto específico de estações de trabalho, sendo ainda cada uma dessas partes, realizada por um usuário diferente.

Cada processo visual define, portanto, um contexto GKS, nele podendo ser utilizadas todas as funções oferecidas pelo GKS e estando a ele associadas várias estações de trabalho.

Neste trabalho, está sendo considerado o fato de que todos os módulos que formam o programa de aplicação exigem apresentação de dados na tela útil, daí a razão do termo processo visual.

É válido salientar que uma mesma estação de trabalho não pode estar associada a dois ou mais processos visuais. Excessão é feita no caso da estação de trabalho pertencente à categoria WISS.

Um termo que será empregado frequentemente neste trabalho, a partir de agora e que deve ser definido neste ponto,

é o de processo visual corrente. Um processo visual é o processo visual corrente se ele estiver sendo executado.

No item seguinte é apresentado o segundo conceito em que se baseia o GAV.

4.2.2- CONFIGURAÇÃO DE TELA

Uma configuração de tela é definida por um conjunto de áreas de visualização independentes e pelo tipo das estações de trabalho que a compõem. Cada uma dessas áreas de visualização é uma estação de trabalho segundo o conceito abstrato de estação de trabalho do GKS.

Os dados das estações de trabalho que compõem uma configuração de tela, podem ser vistos, simultaneamente, na tela útil, enquanto essa configuração de tela for a corrente.

Assim, somente os dados pertencentes às estações de trabalho que compõem a configuração de tela corrente, podem ser vistos pelo usuário final.

É importante salientar que existe uma configuração de tela corrente para cada tipo de estação de trabalho OUTIN definido.

Não é possível apresentar, simultaneamente na tela útil, dados de estações de trabalho que não pertencem à mesma configuração de tela.

É válido ainda salientar que o GAV considera, na composição de configurações de tela, que todas as estações de trabalho pertencem à categoria OUTIN.

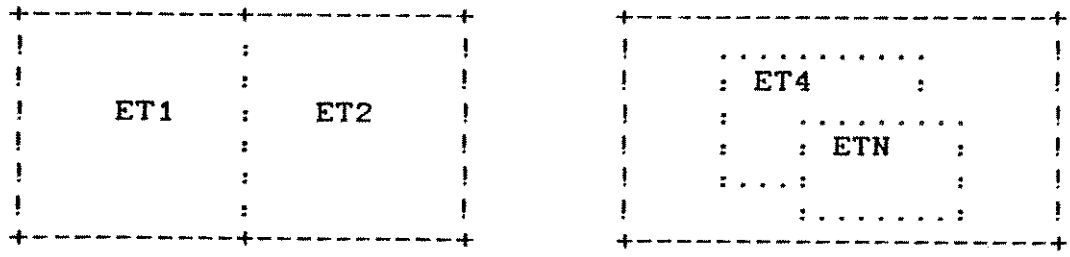

fig. 4.1- Exemplos de configurações de tela

A figura 4.1 mostra exemplos de configurações de tela. Cada uma dessas configurações de tela é formada por uma combinação das N estações de trabalho existentes.

Nessa figura pode-se observar que o GAV permite a superposição das áreas de visualização das estações de trabalho que compõem uma configuração de tela. Neste caso o conteúdo da estação de trabalho de maior prioridade é visto pelo usuário, enquanto a configuração de tela em questão for a configuração de tela corrente.

4.3- VISAO OFERECIDA AOS USUARIOS

O objetivo deste item é dar uma visão geral dos recursos oferecidos pelo GAV aos usuários.

Com a introdução do GAV em um sistema interativo, os usuários vêem um sistema que lhes permite definir diversas estações de trabalho (conceito GKS), vários processos visuais e várias configurações de tela. Essas definições devem constituir a parte inicial do programa de aplicação e devem ser realizadas pelo grupo dos usuários, em um esforço conjunto.

Para proporcionar maior flexibilidade ao trabalho dos usuários, o GAV permite, ainda, a redefinição de estações de trabalho, processos visuais e configurações de tela. Assim, as definições que devem constituir a parte inicial do programa de aplicação, podem ser alteradas pelos diversos módulos que a seguem.

Como já foi dito, a cada processo visual está associado um conjunto de estações de trabalho. Podem pertencer a este conjunto, estações de trabalho pertencentes às categorias: OUTPUT, OUTIN, MO, MI e WISS. Um processo visual pode aplicar os recursos oferecidos pelo pacote gráfico somente ao conjunto de estações de trabalho a ele associadas. Assim, um processo visual envia dados para todas as estações de trabalho ativas, do conjunto de estações de trabalho a ele associadas. No entanto, alterações visuais só serão percebidas pelo usuário final, nas ET's associadas ao processo visual corrente, que façam parte da configuração de tela corrente.

Assim, dependendo da configuração de tela, que esteja definida como corrente, o efeito visual de uma ação pode, ou não, ser percebido pelo usuário final.

Por exemplo, sejam os processos visuais: PV1, ao qual está associada a estação de trabalho ET1 e PV2, ao qual estão associadas as estações de trabalho ET2 e ET3.

Sejam ainda, as configurações de tela: CT1, composta pelas estações de trabalho ET1, ET2 e ET4; e CT2, composta por ET1 e ET3 (figura 4.2).

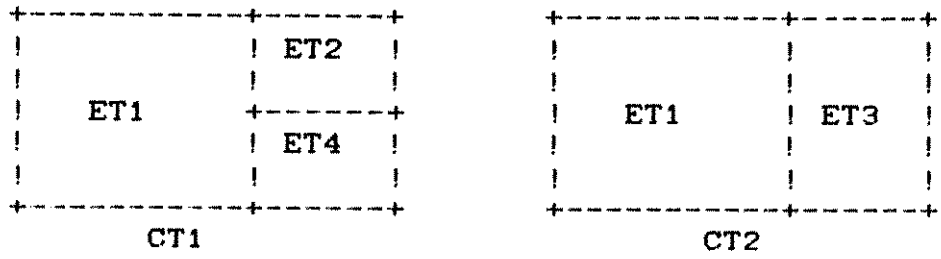

figura 4.2 - configurações de tela CT1 e CT2.

Supondo-se que em um dado instante o processo visual corrente é PV2 e que o conjunto de configurações de tela correntes é formado apenas pela configuração de tela CT2, alterações visuais só serão percebidas pelo usuário na estação de trabalho ET3, pois ET1 não está associada ao processo visual corrente e ET2 não faz parte da configuração de tela corrente.

Quando um processo visual é definido como o processo visual corrente, o ambiente, necessário ao andamento desse novo processo visual, deve ser definido na estrutura de dados do GKS. E, então, resgatado o estado em que se encontrava o GKS, quando o referido processo visual deixou de ser o processo visual corrente. Vê-se então, que o GAV deve dispor de uma estrutura de dados e que nela devem estar armazenadas todas essas informações.

Uma configuração de tela, como já foi visto, é definida por um conjunto de estações de trabalho. Desse conjunto, um subconjunto é composto por estações de trabalho, que estão associadas ao processo visual corrente e o restante das estações de trabalho são ditas externas, pois não estão associadas ao processo visual corrente. Elas contêm os dados necessários para compor, juntamente com as estações de trabalho associadas ao processo visual corrente que fazem parte da configuração de tela

corrente, a tela útil que é apresentada ao usuário final.

Voltando ao exemplo, fornecido neste item e supondo que em um dado instante PV1 é o processo visual corrente e CT1 é a configuração de tela corrente, então, neste instante, ET2 e ET4 são estações de trabalho externas, pois servem apenas para compor a tela útil que é apresentada ao usuário final.

4.4- DEFINIÇÃO FUNCIONAL DO GAV

Como já foi dito, o GAV é um sistema de software que deve permitir a definição de várias estações de trabalho. A cada estação de trabalho OUTIN definida é associada uma fração da tela útil.

Esse sistema deve ainda permitir que os usuários especifiquem as estações de trabalho que devem compor a tela útil em um dado instante, através da definição de configurações de tela.

Além disso, o GAV deve permitir a definição de vários contextos de utilização do pacote gráfico GKS, através da definição de processos visuais. Isso torna possível o projeto e a implementação separada dos módulos que compõem o programa de aplicação.

Devem, portanto, ser alocadas funções no GAV que permitam ao usuário:

1- Definir estações de trabalho

2- Definir processos visuais

3- Definir configurações de tela

4- Definir um processo visual como o processo visual corrente

5- Definir uma configuração de tela como a configuração de tela corrente

6- Utilizar todos os recursos oferecidos pelo pacote gráfico em cada um dos processos visuais e para as estações de trabalho associadas a cada um dos processos

7- Consultar informações da estrutura de dados do GAV

8- Definir prioridades para as estações de trabalho

4.5- SEGMENTOS NO GAV

No GAV, os segmentos possuem dois nomes: um externo, fornecido pelo usuário e outro interno, definido pelo GAV. Lembrando que o GKS não permite a criação de dois segmentos com o mesmo nome, isso permite que os projetistas de diferentes processos visuais, escolham nomes para seus segmentos, sem preocupar-se com aqueles escolhidos pelos demais projetistas.

Assim, os usuários conhecem seus segmentos pelo nome externo, não tendo acesso aos nomes internos dos mesmos.

4.6- ARQUIVOS DE ERRO DO GAV

O GAV oferece aos usuários um arquivo geral de erros. Neste arquivo, o GAV escreve, para cada erro ocorrido durante a execução do programa de aplicação, o identificador de erro, o identificador do processo visual que gerou o erro e a mensagem de erro.

O arquivo geral de erros é aberto pela função do GAV

'OPEN GAV' e é fechado pela função do GAV 'CLOSE GAV'.

Além do arquivo geral de erros, cada processo visual definido dispõe de seu próprio arquivo de erros, cujo nome é fornecido através da função do GAV 'X-OPEN GKS'.

No arquivo de erros de um determinado processo visual, são colocadas informações referentes aos erros, ocorridos somente durante a execução desse processo visual.

4.7- ESTRUTURA DE DADOS DO GAV

Para poder oferecer aos usuários todos os recursos mencionados, o GAV deve dispor de uma estrutura de dados, onde devem estar armazenadas todas as informações necessárias à sua operação.

Essa estrutura de dados é composta por sete tabelas e pode ser vista nas páginas a seguir.

A tabela 4.1 contém informações sobre as estações de trabalho, pertencentes à categoria OUTIN, definidas. Nela são armazenados o identificador da estação de trabalho, o identificador de conexão (utilizado pela interface GKS-DISPOSITIVO para estabelecer a identificação entre o GKS e estações de trabalho do mesmo tipo), seu tipo, sua área de visualização (fração da tela útil que ocupa), sua prioridade, seu índice de cor (utilizado na geração de seu contorno) e área de echo default de seus dispositivos de entrada.

Na tabela 4.2 encontram-se as informações resultantes da definição de estações de trabalho pertencentes às categorias: OUTPUT, METAFILE OUTPUT (MO), METAFILE INPUT (MI) e WORKSTATION

INDEPENDENT SEGMENT STORAGE (WISS).

É importante salientar que o GAV, de acordo com a norma GKS, permite a definição de apenas uma estação de trabalho pertencente à categoria WISS.

A tabela 4.3 contém informações referentes às várias configurações de tela definidas. Nela são armazenados: o identificador da configuração de tela, o tipo e os identificadores das estações de trabalho que a compõem.

Na tabela 4.4 encontram-se as informações referentes aos vários processos visuais definidos. Estão nela contidos: o identificador do processo visual, os identificadores das estações de trabalho associadas a ele, os atributos correntes das primitivas de saída referentes a cada processo visual, o número da transformação de normalização corrente, o estado do GKS para o processo visual, a lista de transformações de normalização por ordem de prioridade, os identificadores das estações de trabalho ativas associadas a ele, o nome do arquivo de erro e o estado da estação de trabalho pertencente à categoria WISS.

A partir das informações contidas nesta tabela, a estrutura de dados do GKS é restabelecida, quando um novo processo visual passa a ser o processo visual corrente.

A tabela 4.5 contém os dados referentes às estações de trabalho abertas. Nela estão contidos: o identificador da estação de trabalho aberta e os nomes externos e internos dos segmentos a ela associados.

A tabela 4.6 contém a lista de segmentos de cada PV que encontram-se armazenados no depósito.

A tabela 4.7 contém os valores correntes necessários ao funcionamento do GAV. Nela estão contidos: o estado corrente do GAV (aberto ou fechado), o identificador do processo visual corrente, os identificadores das configurações de tela correntes (um para cada tipo de ET OUTIN) , a quantidade de transformações de normalização (TN) reais definidas, a quantidade de estações de trabalho abertas, o número utilizado para nome interno de segmento, o flag de erro e o nome do arquivo geral de erros.

identificador da ET	identificador de conexão	tipo	área de visualização	prioridade	indice de cor	área de echo default
1	1	1	(x1,y1) (x2,y2)	1	1	(x3,y3) (x4,y4)
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

TABELA 4.1 - Estações de trabalho OUTIN

identificador da ET	tipo	identificador de conexão	categoria
4	3	2	MO
.	.	.	.
.	.	.	.
.	.	.	.

TABELA 4.2- Estações de trabalho MO, MI, OUTPUT ou WISS

identificador da CT	tipo	estações de trabalho
1	1	1,2,3
.	.	.
.	.	.
.	.	.

TABELA 4.3- Configurações de tela

identificador do PV	ET's associadas	ET's ativas associadas	atributos correntes das prim. de saída	TN corr	estado do GKS p/ o PV	lista de TN	arquivo de erro	estado da ET WISS
1	1,2,3	2,3	linetype linewidth .	1	GKOP	1 2 .	file1	aberta
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

TABELA 4.4- Processos visuais

identificador da ET aberta	lista de segmentos (ext,int)
1	4,1 6,2 .
.	.
.	.

TABELA 4.5- Estações de trabalho abertas

identificador do PV	lista de seg. na WISS
1	3
	4
	.
.	.
.	.
.	.

TABELA 4.6- Segmentos dos PV's na ET WISS

estado do GAV (aberto ou fechado)	
identificador do PV corrente	
identificadores das CT correntes	
quantidade de ET abertas	
quantidade de TN reais definidas	
número utilizado para nome interno de segmento	
flag de erro	
nome do arquivo geral de erros	

TABELA 4.7- Valores correntes

5- PROJETO DE IMPLEMENTAÇÃO DO GAV

Neste capítulo, o comportamento de cada uma das funções oferecidas pelo GAV é descrito. Para isso será utilizada uma linguagem para especificação de projetos que possui estruturas de controle semelhantes às oferecidas pela linguagem Pascal. O apêndice A fornece a descrição das funções do GAV, de uma forma semelhante àquela utilizada pela norma GKS, para descrever as suas funções. A implementação do GAV é descrita pormenorizadamente em /HEL-86/.

A seguir serão detalhados os módulos que compõem o GAV, os quais estão funcionalmente descritos no apêndice A.

5.1- INICIALIZAÇÃO E TRATAMENTO DE ERROS

O GAV oferece ao usuário duas funções, 'OPEN GAV' e 'CLOSE GAV', que estão relacionadas com a inicialização da estrutura de dados do GAV e com o tratamento de erros. O comportamento destas duas funções é mostrado a seguir.

```
OPEN GAV

 IF(GAV já está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF(nome do arquivo geral de erros é inválido)
  THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 .atualizar estado de operação do GAV para "ABERTO"
 .armazenar  nome  do  arquivo geral de erros na  tabela  4.7  da
  estrutura de dados do GAV
```

```
.abrir arquivo geral de erros
.chamar a função do GKS 'OPEN GKS', utilizando o nome do arquivo
 geral de erros como parâmetro de entrada
.inicializar estrutura de dados do GAV
RETURN


CLOSE GAV

 IF(GAV não está aberto)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF(estado do GKS para algum processo visual é diferente de GKCL)
   THEN
     .enviar mensagem de erro
   ELSE
     .fechar arquivo geral de erros
     .atualizar estado de operação do GAV para "FECHADO"
     .chamar a função do GKS 'CLOSE GKS'
 ENDIF
 RETURN
```

5.2- DEFINIÇÃO DE ESTAÇÕES DE TRABALHO

A norma GKS define seis categorias de estações de trabalho: INPUT, OUTPUT, OUTIN, MO, MI e WISS. O GAV permite a definição de estações de trabalho pertencentes apenas às cinco últimas categorias.

Definir uma estação de trabalho, para aquelas pertencentes à categoria OUTIN, significa especificar seu identificador, seu identificador de conexão, sua área de visualização, sua prioridade e seu índice de cor. Para as demais categorias, definir uma estação de trabalho significa especificar seu identificador, seu identificador de conexão e sua categoria.

O GAV deve oferecer uma função que permita ao usuário definir as estações de trabalho de seu interesse. Essa função,

'DEFINE ET', deve, dependendo da categoria da estação de trabalho a ser definida, acessar a tabela 4.1 ou 4.2 da estrutura de dados do GAV. O comportamento dessa função é mostrado a seguir.

É importante observar que a redefinição de uma ET OUTIN, pertencente à configuração de tela corrente, ocasiona a sua atualização.

```
DEFINE ET

 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (categoria não é aceita pelo GAV)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (identificador da ET não é válido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (tipo é inválido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (identificador de conexão não é válido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (área de visualização da ET está fora da tela útil)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (prioridade não é válida)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (índice de cor é inválido)
   THEN
      .enviar mensagem de erro
      RETURN
```

```
ENDIF
IF (categoria é OUTIN)
  THEN
    IF (redefinição)
      THEN
        IF (tipo ou identificador de conexão não estão de acordo
            com os valores armazenados)
          THEN
            .enviar mensagem de erro
            RETURN
        ENDIF
        .atualizar  área de visualização da ET na tabela 4.1  da
         estrutura de  dados  do GAV
        IF (ET aberta)
          THEN
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que atualiza a área de visualização da ET
             na sua estrutura de dados
            .chamar a função do GKS 'SET WORKSTATION VIEWPORT'
            IF (ET pertence à CT corrente)
              THEN
                .limpar a tela útil
                .chamar,  através  de função  'ESCAPE' do  GKS,
                 a   função  do   driver  que   inicializa  o
                 apontador do arquivo de tela da ET especificada
                 (no caso, a ET em questão)
                .chamar  a função do GKS 'REDRAW ALL SEGMENTS ON
                 WORKSTATION'
                .chamar,  através de função 'ESCAPE' do  GKS,  a
                 função  do  driver que regenera o conteúdo das
                 ET's  especificadas (no caso,  as ET's  abertas
                 pertencentes  à  CT  corrente,  exceto  a  ET
                 redefinida)
            ENDIF
        ENDIF
      ELSE
        IF(número  máximo de ET's OUTIN que podem ser  definidas
           já foi atingido)
          THEN
            .enviar mensagem de erro
            RETURN
        ENDIF
       .armazenar   identificador  da   ET,   identificador   de
        conexão, tipo, área de visualização, prioridade e índice
        de cor na tabela 4.1 da estrutura de dados do GAV
    ENDIF
    .inicializar área de echo default
  ELSE
    IF  (existe  ET definida com o  mesmo identificador  ou  com
         mesmo tipo e identificador de conexão)
       THEN
         .enviar mensagem de erro
         RETURN
    ENDIF
    IF(número máximo de ET's MO,  MI,  OUTPUT ou WISS que  podem
```

```
        ser definidas já foi atingido)
         THEN
           .enviar mensagem de erro
           RETURN
     ENDIF
     IF (categoria é WISS)
       THEN
         IF (já existe ET pertencente à categoria WISS)
           THEN
             .enviar mensagem  de erro
             RETURN
         ENDIF
     ENDIF
     .armazenar  identificador da ET,  identificador de  conexão,
      tipo  e categoria na tabela 4.2 da  estrutura de  dados  do
      GAV
ENDIF
RETURN
```

5.3- DEFINIÇÃO DE PROCESSOS VISUAIS

Definir um processo visual significa especificar as estações de trabalho a ele associadas.

O GAV deve oferecer uma função que permita a definição dos diversos processos visuais de interesse.

O comportamento desta função, referida como 'DEFINE PV', é apresentado abaixo.

É importante observar, que o GAV não permite a associação de uma mesma estação de trabalho a dois ou mais processos visuais, exceto se a estação de trabalho em questão pertencer à categoria WISS.

```
DEFINE PV

 IF (GAV não está aberto)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (identificador do PV é inválido)
```

```
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (numero de ET's associadas é maior que o máximo permitido)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
DO  WHILE  (conjunto de ET's associadas ao PV não  completamente
            esgotado)
  IF (ET não definida)
    THEN
      .enviar mensagem de erro
      RETURN
    ELSE
      IF  (ET  não pertence à categoria WISS  e já  se  encontra
           associada a outro PV)
        THEN
          .enviar mensagem de erro
          RETURN
       ENDIF
  ENDIF
END DO
IF (redefinição)
  THEN
    IF (estado do GKS p/ o PV é SGOP)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    .formar  um  conjunto  das  ET's  que  não  mais  se
     encontram associadas ao processo visual em questão
    .atualizar o conjunto de ET's associadas ao  processo visual
     em questão na tabela 4.3 da estrutura de dados do GAV
    DO  WHILE (conjunto de ET's ativas que não se encontram mais
              associadas ao PV  corrente)
      IF (PV é o PV corrente)
        THEN
          .chamar a função do GKS 'DEACTIVATE WORKSTATION'
      ENDIF
      IF (ET pertence à categoria WISS)
         THEN
           .atualizar o estado da ET  WISS para o PV em questão
      ENDIF
      .retirar  identificador  da ET do  conj.  de  ET's  ativas
       associadas ao PV em questão
    END DO
    .formar, a partir do conjunto de ET's abertas, que não estão
     mais associadas ao PV em questão,  n conjuntos: um contendo
     as  ET's que não fazem parte de nenhuma das CT's  correntes
     (conj.  1)  e  um  para cada tipo de  ET  OUTIN  permitido,
     contendo  as ET's que fazem parte da CT corr.  para  aquele
     tipo (conjuntos 2 a n)
    DO WHILE (conjunto 1 não completamente  esgotado)
```

```
    IF (ET pertence à categoria WISS)
      THEN
        .atualizar estado da ET WISS para "CLOSED"
        IF  (ET  não está associada a nenhum outro PV  ou  não
             está  aberta  para  nenhum outro PV)
          THEN
            .chamar a função do GKS  'CLOSE WORKSTATION'
        ENDIF
      ELSE
        .chamar  a função do GKS  'CLOSE WORKSTATION'
        .retirar  o  identificador  da ET  da  tabela  4.5  da
         estrutura de dados do GAV
        IF (ET é OUTIN)
          THEN
            .chamar,  através  de  função 'ESCAPE' do  GKS,  a
             função  do driver que retira o  identificador  de
             conexão  da  ET da sua estrutura de dados
        ENDIF
    ENDIF
  END DO
  DO WHILE (conjuntos 2 a n)
   IF (não há superposição)
     THEN
      DO WHILE (conj. em questão não completamente esgotado)
        .chamar a função do GKS 'CLOSE  WORKSTATION'
        .retirar  identificador  da  ET  da  tabela  4.5  da
         estrutura de dados do GAV
        .chamar,  através de função 'ESCAPE' do GKS, a função
         do  driver  que retira o identificador da ET da  sua
         estrutura de dados
      END DO
     ELSE
       .apagar a tela útil
       .chamar, através de função 'ESCAPE' do GKS, a função do
        driver  que seta o flag de controle do  apagamento  da
        área de visualização da ET na sua estrutura de dados
       DO WHILE (conj. em questão não completamente esgotado)
         .chamar a função do GKS 'CLOSE WORKSTATION'
         .retirar  o  identificador  da ET da  tabela  4.5  da
          estrutura de dados do GAV
         .chamar,  através de função 'ESCAPE' do GKS, a função
          do  driver que retira o  identificador de conexão da
          ET em questão da sua estrutura de dados
       END DO
       .chamar, através de função 'ESCAPE' do GKS, a função do
        driver  que  regenera o conteúdo das ET's do  conjunto
        especificado  (neste  caso,  todas  as  ET's  abertas
        pertencentes à CT corrente)
       .chamar, através de função 'ESCAPE' do GKS, a função do
        driver que reseta o flag de controle do  apagamento da
        área de visualização  da ET
   ENDIF
  END DO
ELSE
  IF  (número  máximo  de PV que podem ser  definidos  já  foi
```

```
          atingido)
        THEN
          .enviar mensagem de erro
          RETURN
    ENDIF
    .armazenar  o  identificador  do  PV,  juntamente  com  os
     identificadores  das  ET's  associadas,  na tabela  4.4  da
     estrutura de dados do GAV
    .inicializar a tabela 4.4 da estrutura de dados do GAV
ENDIF
RETURN
```

5.4- DEFINIÇÃO DE CONFIGURAÇÕES DE TELA

Definir uma configuração de tela significa especificar as estações de trabalho que a compõem. A função do GAV responsável por essa tarefa chama-se 'DEFINE CT'.

O comportamento desta função é descrito a seguir.

```
DEFINE CT

 IF (GAV não está aberto)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (identificador da CT não é válido)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF  (número  de  ET's  pertencentes à CT é maior  que  o  máximo
      permitido)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 DO  WHILE  (conjunto de ET's candidatas a compor a CT ainda  não
             completamente esgotado)
   IF  (ET não definida ou não pertencente à categoria  OUTIN  ou
        existem tipos distintos de ET's)
     THEN
       .enviar mensagem de erro
       RETURN
   ENDIF
 END DO
```

```
IF (redefinição)
  THEN
    .formar  um conjunto contendo as ET's que passaram  a  fazer
     parte  da  CT  (conjunto  A  )  e  outro  contendo  as  que
     continuaram pertencendo à CT em questão (conjunto B)
    .atualizar  conjunto de ET's que compõem a CT em questão  na
     tabela 4.3 da estrutura de dados do GAV
    IF (CT é  CT corrente)
      THEN
        .chamar,  através de função 'ESCAPE' do GKS, a função do
         driver  que atualiza o conjunto das ET's que compõem  a
         CT corrente
        .limpar a tela útil
        DO  WHILE (subconjunto de ET's abertas do conjunto A não
                   esgotado)
          .chamar a função do GKS 'UPDATE WORKSTATION',  tendo o
           flag  de  regeneração o valor  SUPPRESS  (desbloqueia
           ações adiadas)
        END DO
        .limpar a tela útil
        .chamar,  através de função 'ESCAPE' do GKS, a função do
         driver  que  seta o flag de controle do  apagamento  da
         área de visualização da ET, na sua estrutura de dados
        DO WHILE (subconjunto de ET's abertas do conjunto A  não
                  completamente esgotado)
          .chamar,  através de função ESCAPE do GKS, a função do
           driver que inicializa o apontador do arquivo de  tela
           da ET em questão
          .chamar  a  função  do  GKS 'REDRAW  ALL  SEGMENTS  ON
           WORKSTATION'
        END DO
        .chamar,  através de função 'ESCAPE' do GKS, a função do
         driver  que regenera o conteúdo da  ET's  especificadas
         (neste caso,  o subconjunto de ET's abertas do conjunto
         B)
        .chamar,  através de função 'ESCAPE' do GKS, a função do
         driver  que reseta o flag de controle do apagamento  da
         área de visualização da ET na sua estrutura de dados
    ENDIF
  ELSE
    IF  (número  máximo  de CT que podem ser  definidas  já  foi
         atingido)
       THEN
         .enviar mensagem de erro
         RETURN
    ENDIF
    .armazenar   o  identificador  da  CT,   juntamente  com  os
     identificadores  das  ET's que a compõem na tabela  4.3  da
     estrutura de dados do GAV
ENDIF
RETURN
```

5.5- DEFINIÇÃO DE PROCESSO VISUAL CORRENTE

O GAV deve oferecer uma função ao usuário para que ele possa definir o processo visual corrente.

O comportamento desta função, referida como 'SET PV CORRENTE', é apresentado abaixo.

```
SET PV CORRENTE

 IF (GAV não está aberto)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (identificador do PV é inválido)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (PV não foi definido)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é SGOP)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (PV já é o PV corrente)
   THEN
     RETURN
 ENDIF
 .desativar ET's cujos identificadores encontrem-se no conjunto
  de ET's ativas associadas ao PV corrente, na tabela 4.4 da
  estrutura de dados do GAV
 .atualizar a entrada 'processo visual corrente' da tabela 4.7 da
  estrutura de dados do GAV
 .copiar dados relativos ao PV corrente da tabela 4.4 da
  estrutura de dados do GAV na estrutura de dados do GKS
 .ativar ET's pertencentes ao conjunto de ET's ativas associadas
  ao novo PV corrente
 RETURN
```

5.6- DEFINIÇÃO DE CONFIGURAÇÃO DE TELA CORRENTE

Uma função do GAV deve ser responsável pela definição da configuração de tela corrente. Essa função chama-se 'DEFINE CT CORRENTE' e seu comportamento é mostrado a seguir.

```
DEFINE CT CORRENTE

 IF (GAV não está aberto)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (identificador da CT é inválido)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (CT não foi definida)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (CT em questão já é a CT corrente)
   THEN
     RETURN
 ENDIF
 .atualizar entrada 'configuração de tela corrente', para o tipo
  em questão na tabela 4.7 da estrutura de dados do GAV
 IF (ao menos uma das ET's pertencentes à antiga CT corrente está
     aberta)
   THEN
     .limpar a tela útil
     RETURN
 ENDIF
 IF  (ao menos uma das ET's que compõem a nova CT corrente está
      aberta)
   THEN
     DO  WHILE  (conjunto de ET's abertas pertencentes à nova CT
                 corrente não esgotado)
       .chamar a função do GKS 'UPDATE WORKSTATION', tendo o flag
        de regeneração o valor 'SUPPRESS'( desbloqueia ações
        adiadas)
     END DO
     .limpar a tela útil
     .chamar, através de função 'ESCAPE' do GKS, a função do
      driver que seta o flag de controle do apagamento da área de
      visualização da ET na sua estrutura de dados
     DO  WHILE (conjunto de ET's abertas pertencentes à CT
```

```
                corrente em questão não completamente esgotado)
        .chamar,  através de função 'ESCAPE' do GKS,  a função  do
        driver  que inicializa o apontador do arquivo de tela  da
        ET em questão
        .chamar   a   função   do   GKS   'REDRAW   ALL   SEGMENTS   ON
        WORKSTATION'
    END DO
    .chamar,  através  de função 'ESCAPE' do GKS,  a  função  do
     driver  que reseta o flag de controle do apagamento da área
     de visualização da ET na sua estrutura de dados
ENDIF
RETURN
```

5.7- DEFINIÇÃO DA PRIORIDADE DE ESTAÇÕES DE TRABALHO

Uma vez que o GAV permite a superposição de áreas de visualização de estações de trabalho, deve ser oferecida ao usuário, uma função que permita a definição de uma ordem de prioridade entre as mesmas. Essa função, 'SET PRIORIDADE', deve apresentar o seguinte comportamento:

```
SET PRIORIDADE

 IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
    RETURN
 ENDIF
 IF (ET não está associada ao PV corrente)
       THEN
          .enviar mensagem de erro
          RETURN
 ENDIF
 IF (ET não pertence à categoria OUTIN)
   THEN
      .enviar mensagem de erro
     RETURN
 ENDIF
 IF (prioridade não é válida)
   THEN
      .enviar mensagem de erro
     RETURN
 ENDIF
 IF (ET está aberta)
   THEN
```

```
     .chamar, através de função 'ESCAPE' do GKS, a função do
      driver que atualiza a prioridade da ET especificada (no
      caso, a ET em questão)
     IF (ET pertence à CT corrente)
       THEN
          .analisar prioridade
          IF (regeneração é necessária)
             THEN
                .limpar a tela útil
                .chamar, através de função 'ESCAPE' do GKS, a
                 função dodriver que regenera o conteúdo das ET's
                 especificadas (no caso, todas as ET's abertas
                 pertencentes à CT corrente)
          ENDIF
     ENDIF
ENDIF
.atualizar a prioridade da ET em questão na tabela 4.1 da
 estrutura de dados do GAV
RETURN
```

5.8- UTILIZAÇÃO DOS RECURSOS GRAFICOS

Como já foi dito, os usuários devem poder utilizar todos os recursos oferecidos pelo GKS em cada um dos processos visuais e para cada uma das estações de trabalho.

Para que isso seja possível, é necessário definir uma função do GAV para cada função oferecida pelo GKS.

Assim, foram criadas funções GAV, denominadas da mesma forma que as funções do GKS, acrescidas de um X à frente. O comportamento destas funções é descrito a seguir e sua descrição funcional é a mesma apresentada na norma GKS.

5.8.1- FUNÇOES DE CONTROLE

```
X-OPEN GKS

 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente não é GKCL)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (nome do arquivo de erros não é válido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF ( nome de arquivo já utilizado por outro PV)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 .atualizar  estado do GKS para o PV corrente,  na tabela 4.4  da
  estrutura de dados do GAV
 .abrir arquivo de erros do processo visual corrente
 .armazenar  nome do arquivo de erros na tabela 4.4 da  estrutura
  de dados do GAV, para o PV corrente
 RETURN
```

X-CLOSE GKS

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente não é GKOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
.atualizar  estado  do GKS para o PV corrente na tabela  4.4  da
 estrutura de dados do GAV
.fechar arquivo de erros do PV corrente
RETURN
```

X-OPEN WORKSTATION

```
IF (GAV  não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL)
  THEN
    enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF  (identificador de conexão e tipo não são iguais aos  valores
     armazenados)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (número máximo de ET's abertas já foi atingido)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET já está aberta)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET pertence à categoria OUTIN)
  THEN
    .chamar,  através  de função 'ESCAPE' do GKS,  a  função  do
```

```
    driver que armazena dados da ET no driver
  IF (é a primeira ET OUTIN de seu tipo a ser aberta)
    THEN
      .chamar, através de função 'ESCAPE' do GKS, a função do
       driver que inicializa o dispositivo
  ENDIF
  IF (ET pertence à CT corrente)
    THEN
      IF (não há superposição)
        THEN
          .chamar a função do GKS 'OPEN WORKSTATION'
          .chamar função do GKS 'SET WORKSTATION VIEWPORT'
        ELSE
          .apagar a tela útil
          .chamar, através de função 'ESCAPE' do GKS, a função
           do driver que seta o flag de controle do apagamento
           da área de visualização da ET
          .chamar a função do GKS 'OPEN WORKSTATION'
          .chamar a função do GKS 'SET WORKSTATION VIEWPORT'
          .chamar, através de função 'ESCAPE' do GKS, a função
           do driver que regenera o conteúdo das ET's
           especificadas (neste caso, demais ET's abertas que
           compõem a CT corrente)
          .chamar, através de função 'ESCAPE'do GKS, a função
           do driver que reseta o flag de controle do
           apagamento da área de visualização da ET na sua
           estrutura de dados
      ENDIF
    ELSE
      .chamar a função do GKS 'OPEN WORKSTATION'
      .chamar a função do GKS 'SET WORKSTATION VIEWPORT'
  ENDIF
  .armazenar o identificador da ET na tabela 4.5 da estrutura
   de dados do GAV
  .inicializar dispositivos de entrada
ELSE
  IF (ET pertence à categoria WISS)
    THEN
      IF (estado da ET WISS para o PV corrente é 'aberta')
        THEN
          .enviar mensagem de erro
          RETURN
      ENDIF
      IF (estado da ET WISS para o PV corrente é 'ativa')
        THEN
          .enviar mensagem de erro
          RETURN
      ENDIF
      IF (ET ainda não aberta)
        THEN
          .chamar a função do GKS 'OPEN WORKSTATION'
      ENDIF
      .atualizar o estado da ET WISS para o PV corrente para
       'aberta'
    ELSE
```

```
          .chamar a função do GKS 'OPEN WORKSTATION'
          .colocar o identificador da ET na tabela 4.5 da
           estrutura de dados do GAV
      ENDIF
 ENDIF
 IF (estado do GKS para o PV corrente é GKOP)
   THEN
      .atualizar o estado do GKS para o PV corrente para WSOP
 ENDIF
 .incrementar a quantidade de ET's abertas
 RETURN


X-CLOSE WORKSTATION

 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está associada ao PV corrente)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está aberta)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está ativa)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET pertence à categoria OUTIN)
   THEN
      IF (ET pertence à CT corrente)
        THEN
          IF (não há superposição)
            THEN
              .chamar função do GKS 'CLOSE WORKSTATION'
            ELSE
              .chamar, através de função 'ESCAPE' do GKS, a função
               do driver que seta flag de controle do apagamento
               da área de visualização da ET no driver
              .chamar a função do GKS 'CLOSE WORKSTATION'
              .limpar a tela útil
              .chamar, através de função 'ESCAPE'do GKS, a função
```

```
            do driver que regenera o conteúdo das ET's
            especificadas (neste caso, todas as ET's abertas
            pertencentes à CT corrente, exceto a ET em questão)
           .chamar, através de função 'ESCAPE'do GKS, a função
            do driver que reseta o flag de controle do
            apagamento da área de visualização da ET na sua
            estrutura de dados
        ENDIF
      ELSE
        .chamar a função do GKS 'CLOSE WORKSTATION'
    ENDIF
    IF (ET é a última ET OUTIN a ser fechada)
      THEN
        .chamar, através de função 'ESCAPE' do GKS, a função do
         driver que reseta o dispositivo
    ENDIF
    .chamar, através de função 'ESCAPE' do GKS, a função do
     driver que retira, da estrutura de dados do driver, o
     identificador de conexão da ET em questão
    .retirar identificador da ET em questão da tabela 4.5 da
     estrutura de dados do GAV
  ELSE
    IF (ET pertence à categoria WISS)
      THEN
        IF (estado da ET WISS para o PV corrente é 'fechada')
          THEN
            .enviar mensagem de erro
            RETURN
        ENDIF
        IF (estado da ET WISS para o PV corrente é 'ativa')
          THEN
            .enviar mensagem de erro
            RETURN
        ENDIF
        .atualizar o estado da ET WISS para o PV corrente
        IF (ET não está aberta para nenhum outro PV ou não está
             associada a nenhum outro PV)
          THEN
            .chamar a função do GKS 'CLOSE WORKSTATION'
        ENDIF
      ELSE
        .chamar a função do GKS 'CLOSE WORKSTATION'
        .retirar identificador da ET da tabela 4.5 da estrutura
         de dados do GAV
    ENDIF
ENDIF
IF (não existe nenhuma ET aberta associada ao PV corrente)
  THEN
    .atualizar o estado do GKS para o PV corrente na tabela 4.4
     da estrutura de dados do GAV
ENDIF
.decrementar a quantidade de ET's abertas
RETURN
```

X-ACTIVATE WORKSTATION

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP ou SGOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mesagem de erro
    RETURN
ENDIF
IF (ET pertence à categoria WISS)
  THEN
    IF (estado da ET WISS para o PV corrente é 'fechada')
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (estado da ET WISS para o PV corrente é 'ativa')
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    .chamar a função do GKS 'ACTIVATE WORKSTATION'
    .atualizar o estado da ET WISS para o PV corrente para
     'ativa'
  ELSE
    .chamar a função do GKS 'ACTIVATE WORKSTATION'
ENDIF
.armazenar identificador da ET no conjunto de ET's ativas
 associadas ao PV corrente
IF (estado do GKS para o PV corrente é WSOP)
  THEN
    .atualizar o estado do GKS para o PV corrente, na tabela 4.4
     da estrutura de dados do GAV, para WSAC
ENDIF
RETURN
```

X-DEACTIVATE WORKSTATION

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
```

```
IF (estado do GKS para o PV corrente não é WSAC)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET pertence à categoria WISS)
  THEN
     IF (ET não está ativa)
       THEN
          .enviar mensagem de erro
          RETURN
     ENDIF
     .chamar a função do GKS 'DEACTIVATE WORKSTATION'
     .atualizar o estado da ET WISS para o PV corrente para
      'ABERTA'
  ELSE
     .chamar a função do GKS 'DEACTIVATE WORKSTATION'
ENDIF
.retirar identificador da ET do conjunto de ET's ativas
 associadas ao PV corrente
IF (não existe outra ET ativa associada ao PV corrente)
  THEN
     .atualizar estado do GKS para o PV corrente, na tabela 4.4
      da estrutura de dados do GAV, para WSOP
ENDIF
RETURN


X-CLEAR WORKSTATION

IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente não é WSOP ou WSAC)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET não está aberta)
  THEN
     .enviar mensagem de erro
     RETURN
```

```
ENDIF
IF (ET pertence à categoria OUTIN)
  THEN
    IF (ET pertence à CT corrente)
      THEN
        IF (não há superposição)
          THEN
            .chamar a função do GKS 'CLEAR WORKSTATION'
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que inicializa o apontador do arquivo de
             tela da ET
          ELSE
            .limpar a tela útil
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que seta o flag de controle do apagamento
             da área de visualização da ET
            .chamar a função do GKS 'CLEAR WORKSTATION'
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que regenera o conteúdo das ET's
             especificadas (neste caso, ET's abertas
             pertencentes à CT corrente, exceto a ET em questão)
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que inicializa o apontador do arquivo de
             tela da ET
        ENDIF
      ELSE
        .chamar a função do GKS 'CLEAR WORKSTATION'
        .chamar, através de função 'ESCAPE' do GKS, a função do
         driver que inicializa o apontador do arquivo de tela da
         ET
    ENDIF
  ELSE
    .chamar a função do GKS 'CLEAR WORKSTATION'
ENDIF
.esvaziar a lista de segmentos da ET na tabela 4.5 da estrutura
 de dados do GAV
RETURN


X-REDRAW ALL SEGMENTS ON WORKSTATION

IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
```

```
    RETURN
ENDIF
IF (ET pertence à categoria OUTIN)
  THEN
    IF (ET pertence à CT corrente)
      THEN
        IF (não há superposição)
          THEN
            .chamar a função do GKS 'UPDATE WORKSTATION', tendo
             o flag de regeneração o valor 'SUPPRESS'
             (desbloqueia ações adiadas)
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que inicializa o apontador do arquivo de
             tela da ET
            .chamar a função do GKS 'REDRAW ALL SEGMENTS ON
             WORKSTATION'
          ELSE
            .limpar a tela útil
            .chamar a função do GKS 'UPDATE WORKSTATION', tendo
             o flag de regeneração o valor 'SUPPRESS'
            .limpar a tela útil
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que seta o flag de controle do apagamento
             da área de visualização da ET
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que inicializa o apontador do arquivo de
             imagem da ET em questão
            .chamar a função do GKS 'REDRAW ALL SEGMENTS ON
             WORKSTATION'
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que regenera o conteúdo das ET's
             especificadas (neste caso, todas as ET's abertas
             pertencentes à CT corrente, exceto a ET em questão)
            .chamar, através de função 'ESCAPE'do GKS, a função
             do driver que reseta o flag de controle do
             apagamento da área de visualização da ET
        ENDIF
      ELSE
        .chamar a função do GKS 'UPDATE WORKSTATION', tendo o
         flag de regeneração o valor SUPPRESS
        .chamar, através de função 'ESCAPE' do GKS, a função do
         driver que inicializa o apontador do arquivo de tela da
         ET
        .chamar a função do GKS 'REDRAW ALL SEGMENTS ON
         WORKSTATION'
    ENDIF
  ELSE
    .chamar a função do GKS 'REDRAW ALL SEGMENTS ON WORKSTATION'
ENDIF
RETURN
```

```
X-UPDATE WORKSTATION

 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está associada ao PV corrente)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está aberta)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET é OUTIN)
   THEN
      IF ( é apenas desbloqueio de ações adiadas)
        THEN
          .chamar a função do GKS 'UPDATE WORKSTATION'
        ELSE
          IF (ET pertence à CT corrente)
            THEN
              IF (não há superposição)
                THEN
                  .chamar  a função do GKS  'UPDATE  WORKSTATION',
                   tendo  o flag de regeneração o valor 'SUPPRESS'
                   (desbloqueia ações adiadas)
                  .chamar,  através de função 'ESCAPE' do  GKS,  a
                   função  do  driver que inicializa apontador  do
                   arquivo de tela da ET em questão
                  .chamar a função do GKS 'UPDATE WORKSTATION'
                ELSE
                  .limpar a tela útil
                  .chamar  a função do GKS  'UPDATE  WORKSTATION',
                   tendo  o flag de regeneração implícita o  valor
                   'SUPPRESS' (desbloqueia ações adiadas)
                  .limpar a tela útil
                  .chamar,  através  de função 'ESCAPE' do GKS,  a
                   função do driver que seta o flag de controle do
                   apagamento da área de visualização da ET na sua
                   estrutura de dados
                  .chamar,  através  de função 'ESCAPE' do GKS,  a
                   função  do driver que inicializa o apontador do
                   arquivo de imagem da ET em questão
                  .chamar função do GKS 'UPDATE WORKSTATION'
                  .chamar,  através  de função 'ESCAPE' do GKS,  a
                   função  do  driver que regenera o conteúdo  das
```

```
                    ET's especificadas (neste caso, ET's abertas
                    pertencentes à CT corrente, exceto a ET em
                    questão)
                   .chamar, através de função 'ESCAPE' do GKS, a
                    função do driver que reseta o flag de controle
                    do apagamento da área de visualização da ET na
                    sua estrutura de dados
                ENDIF
              ELSE
                .chamar a função do GKS 'UPDATE WORKSTATION', tendo
                 o flag de regeneração o valor 'SUPPRESS' (
                 desbloqueia ações adiadas)
                .chamar, através de função 'ESCAPE' do GKS, a função
                 do driver que inicializa o apontador do arquivo de
                 tela da ET em questão
                .chamar a função do GKS 'UPDATE WORKSTATION'
            ENDIF
        ENDIF
      ELSE
        .chamar a função do GKS 'UPDATE WORKSTATION'
    ENDIF
    RETURN


X-SET DEFERRAL STATE

    IF (GAV não está aberto)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (ET não está associada ao PV corrente)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (ET está fechada)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (ET pertence à categoria OUTIN)
      THEN
        IF  (modo de regeneração implícita é ALLOWED e 'new frame
             action' é YES)
          THEN
            IF (ET pertence à CT corrente)
              THEN
                IF (não há superposição)
```

```
THEN
  IF (modo de adiamento é ASAP)
    THEN
      .chamar a função do GKS 'UPDATE WORKSTATION',
       tendo o flag de regeneração o valor
       'SUPPRESS' (desbloqueia ações adiadas)
  ENDIF
  .chamar, através de função 'ESCAPE' do GKS, a
   função do driver que inicializa o apontador do
   arquivo de tela da ET
  .chamar função do GKS 'SET DEFERRAL STATE'
ELSE
  IF (modo de adiamento é ASAP)
     THEN
       .limpar a tela útil
       .chamar a função do GKS 'UPDATE
        WORKSTATION', tendo o flag de regeneração o
        valor SUPPRESS
       .limpar a tela útil
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que seta o flag de
        controle do apagamento da área de
        visualização da ET
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que inicializa o
        apontador do arquivo de imagem da ET
        especificada (no caso, a ET em questão)
       .chamar a função do GKS 'SET DEFERRAL STATE'
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que regenera o conteúdo
        das ET's especificadas (no caso, demais
        ET's abertas pertencentes à CT corrente)
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que reseta o flag de
        controle do apagamento da área de
        visualização da ET
     ELSE
       .limpar a tela útil
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que seta flag de
        controle do apagamento da área de
        visualização da ET
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que inicializa o
        apontador do arquivo de tela da ET
       .chamar a função do GKS 'SET DEFERRAL STATE'
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que regenera o conteúdo
        das ET's especificadas (no caso, todas as
        ET's abertas pertencentes à CT corrente,
        exceto a ET em questão)
       .chamar, através de função 'ESCAPE' do GKS,
        a função do driver que reseta o flag de
        controle do apagamento da área de
        visualização da ET
```

```
                ENDIF
              ENDIF
            ELSE                /* ET não pert. à CT corr. */
              IF (modo de adiamento é ASAP)
                THEN
                  .chamar a função do GKS 'UPDATE WORKSTATION',
                   tendo o flag de regeneração o valor SUPPRESS
                   (desbloqueia ações adiadas)
              ENDIF
              .chamar, através de função 'ESCAPE' do GKS, a função
               do driver que inicializa o apontador do arquivo de
               tela da ET
              .chamar a função do GKS 'SET DEFERRAL STATE'
          ENDIF
        ELSE                 /* não vai haver regeneração */
          .chamar a função do GKS 'SET DEFERRAL STATE'
      ENDIF
    ELSE                    /* ET não é OUTIN */
      .chamar a função do GKS 'SET DEFERRAL STATE'
  ENDIF
  RETURN


X-MESSAGE


  IF (GAV não está aberto)
    THEN
      .enviar mensagem de erro
      RETURN
  ENDIF
  IF (estado do GKS para o PV corrente é GKCL ou GKOP)
    THEN
      .enviar mensagem de erro
      RETURN
  ENDIF
  IF (ET não está associada ao PV corrente)
    THEN
      .enviar mensagem de erro
      RETURN
  ENDIF
  .chamar a função do GKS 'MESSAGE'
  RETURN
```

X-ESCAPE

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV é GKCL)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
.chamar a função do GKS 'ESCAPE'
RETURN
```

5.8.2- FUNÇOES DE SAIDA

São as seguintes, as funções de saída do GAV:

X-POLYLINE,

X-POLYMARKER,

X-TEXT,

X-FILL AREA,

X-CELL ARRAY e

X-GENERALIZED DRAWING PRIMITIVE.

Cada uma destas funções se comporta da seguinte forma:

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP ou WSOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é SGOP)
  THEN
    DO WHILE (conjunto de ET's ativas não esgotado)
      IF (ET pertence à CT corrente e vai ocorrer regeneração do
          conteúdo da ET)
        THEN
          IF (não há superposição)
            THEN
```

```
                .chamar, através de função 'ESCAPE' do GKS, a
                 função do driver que inicializa o apontador do
                 arquivo de imagem da ET
            ELSE
              .flag=1
              .armazenar no conjunto A o identificador da CT
               corrente em questão
              .chamar, através de função 'ESCAPE' do GKS, a
               função do driver que inicializa o apontador do
               arquivo de imagem da ET
          ENDIF
        ENDIF
      END DO
      .chamar a função do GKS correspondente
      IF (flag é igual a 1)
        THEN
          DO WHILE (conjunto A não esgotado)
            .apagar a tela útil
            .regenerar conteúdo das ET's abertas que a compõem
          END DO
      ENDIF
    ELSE
      .chamar a função do GKS correspondente
  ENDIF
  RETURN
```

5.8.3- FUNÇOES QUE DEFINEM ATRIBUTOS

As funções do GAV:

X-SET POLYLINE INDEX,

X-SET LINETYPE,

X-SET LINEWIDTH SCALE FACTOR,

X-SET POLYLINE COLOUR INDEX,

X-SET POLYMARKER INDEX,

X-SET MARKER TYPE,

X-SET MARKER SIZE SCALE FACTOR,

X-SET POLYMARKER COLOUR INDEX,

X-SET TEXT INDEX,

X-SET TEXT FONT AND PRECISION,

X-SET CHARACTER EXPANSION FACTOR,

X-SET CHARACTER SPACING,

X-SET TEXT COLOUR INDEX,

X-SET CHARACTER HEIGHT,

X-SET CHARACTER UP VECTOR,

X-SET TEXT PATH,

X-SET TEXT ALIGNMENT,

X-SET FILL AREA INDEX,

X-SET FILL AREA INTERIOR STYLE,

X-SET FILL AREA STYLE INDEX,

X-SET FILL AREA COLOUR INDEX,

X-SET PATTERN SIZE,

X-SET PATTERN REFERENCE POINT,

X-SET ASPECT SOURCE FLAGS,

X-SET PICK IDENTIFIER,

possuem o comportamento mostrado abaixo.

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
  ELSE
    IF (estado do GKS para o PV corrente é GKCL)
       THEN
         .enviar mensagem de erro
       ELSE
         .chamar a função do GKS correspondente
         IF (flag de erro é igual a zero)
           THEN
             .atualizar  valor do atributo em questão na  tabela
              4.4 da estrutura de dados do GAV
         ENDIF
    ENDIF
ENDIF
RETURN
```

5.8.4- FUNÇOES QUE DEFINEM AS REPRESENTAÇOES

As funções do GAV:

X-SET POLYLINE REPRESENTATION,

X-SET POLYMARKER REPRESENTATION,

X-SET TEXT REPRESENTATION,

X-SET FILL AREA REPRESENTATION,

X-SET PATTERN REPRESENTATION,

X-SET COLOUR REPRESENTATION,

possuem o comportamento mostrado abaixo.

```
 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está associada ao PV corrente)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF  (ET  pertence  à CT corrente e vai  ocorrer  regeneração  do
      conteúdo da ET)
   THEN
      IF (não há superposição)
        THEN
          .inicializar  o  apontador do  arquivo  de imagem da ET
          .chamar a função do GKS correspondente
        ELSE
          .apagar a tela útil
          .chamar,  através  de função 'ESCAPE' do GKS a função do
           driver que seta o flag de controle do apagamento da ET
          .chamar,  através de função 'ESCAPE' do GKS, a função do
           driver que inicializa o apontador do arquivo imagem da ET
          .chamar a função do GKS correspondente
          .chamar,  através de função 'ESCAPE' do GKS, a função do
           driver  que regenera o conteúdo das ET's  especificadas
           (no  caso,  as  demais ET's abertas pertencentes  à  CT
           corrente)
          .chamar,  através de função 'ESCAPE' do GKS, a função do
           driver  que reseta o flag de controle do apagamento  da
```

```
          área de visualização da ET
      ENDIF
    ELSE
      .chamar a função do GKS correspondente
 ENDIF
 RETURN
```

5.8.5- FUNÇOES RELACIONADAS COM AS TRANSFORMAÇOES

As funções do GAV:

X-SET WINDOW e

X-SET VIEWPORT,

apresentam o comportamento abaixo.

```
 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (número virtual da TN é inválido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (número virtual da TN é zero)
   THEN
      .chamar função do GKS correspondente, utilizando como
       parâmetro de entrada a TN número zero
   ELSE
      .obter número real da TN (ver tabelas 4.4 e 4.7)
      .chamar a função do GKS correspondente, utilizando o número
       real da TN como parâmetro de entrada
 ENDIF
 RETURN
```

X-SET VIEWPORT INPUT PRIORITY

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL)
 THEN
   .enviar mensagem de erro
   RETURN
ENDIF
IF (algum dos números de TN é inválido)
  THEN
    .enviar mensagem de erro
  ELSE
    .atualizar  a  ordem das TN na tabela 4.4  da  estrutura  de
     dados do GAV
    .chamar a função do GKS 'SET VIEWPORT INPUT PRIORITY'
ENDIF
RETURN
```

X-SELECT NORMALIZATION TRANSFORMATION

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (número virtual da TN é inválido)
  THEN
    .enviar mensagem de erro
  ELSE
    .obter número real da TN
    .chamar    a    função    do    GKS    'SELECT    NORMALIZATION
    TRANSFORMATION'
    .atualizar  o número da TN corrente para o PV  corrente,  na
     tabela 4.4 da estrutura de dados do GAV
ENDIF
RETURN
```

X-SET CLIPPING INDICATOR

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.chamar função do GKS 'SET CLIPPING INDICATOR'
.atualizar  entrada 'indicador de clipping' para o PV  corrente,
 na tabela 4.4 da estrutura de dados do GAV
ENDIF
RETURN
```

X-SET WORKSTATION WINDOW

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF  (ET  pertence  à CT corrente e vai  ocorrer  regeneração  do
     conteúdo da ET)
   THEN
     IF (não há superposição)
       THEN
          .chamar através de função 'ESCAPE' do GKS,  a função do
           driver que inicializa o apontador do arquivo imagem da
           ET
          .chamar a função do GKS 'SET WORKSTATION WINDOW'
       ELSE
          .apagar a tela útil
          .chamar,  através de função 'ESCAPE' do GKS a função do
           driver  que seta o flag de controle do  apagamento  da
           área de visualização da ET
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que inicializa o apontador do arquivo imagem da
```

```
            ET
           .chamar a função do GKS 'SET WORKSTATION WINDOW'
           .chamar através de função 'ESCAPE' do GKS,  a função do
            driver  que regenera o conteúdo das ET's especificadas
            (no  caso,  as demais ET's abertas pertencentes  à  CT
            corrente)
           .chamar, através de função 'ESCAPE' do GKS, a função do
            driver  que reseta o flag de controle do apagamento da
            área de visualização da ET
      ENDIF
    ELSE
      .chamar a função do GKS 'SET WORKSTATION WINDOW'
ENDIF
RETURN


X-SET WORKSTATION VIEWPORT


IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF ("viewport" não está dentro da área de visualização da ET)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF  (ET  pertence  à CT corrente e vai  ocorrer  regeneração  do
     conteúdo da ET)
  THEN
    IF (não há superposição)
      THEN
        .chamar,  através de função 'ESCAPE' do GKS, a função do
         driver que inicializa o apontador do arquivo imagem  da
         ET
        .chamar a função do GKS 'SET WORKSTATION VIEWPORT'
      ELSE
        .apagar a tela útil
        .chamar,  através de função 'ESCAPE' do GKS, a função do
         driver  que  seta o flag de controle do  apagamento  da
         área de visualização da ET
        .chamar,  através de função 'ESCAPE' do GKS, a função do
```

```
            driver  que inicializa o apontador do arquivo imagem da
            ET
           .chamar a função do GKS 'SET WORKSTATION VIEWPORT'
           .chamar,  através de função 'ESCAPE' do GKS, a função do
            driver  que regenera o conteúdo das ET's  especificadas
            (no  caso,  as  demais ET's abertas pertencentes  à  CT
             corrente)
           .chamar,  através de função 'ESCAPE' do GKS, a função do
            driver  que reseta o flag de controle do apagamento  da
            área de visualização da ET
      ENDIF
    ELSE
      .chamar função do GKS 'SET WORKSTATION   VIEWPORT'
 ENDIF
 RETURN
```

5.8.6- FUNÇOES RELACIONADAS COM SEGMENTOS

5.8.6.1- DEFINIÇAO DE ATRIBUTOS

As funções do GAV:

X-SET SEGMENT TRANSFORMATION,

X-SET VISIBILITY,

X-SET HIGHLIGHTING,

X-SET SEGMENT PRIORITY,

possuem o comportamento mostrado a seguir.

```
 IF (GAV não está aberto)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (nome do segmento é inválido)
   THEN
     .enviar mensagem de erro
     RETURN
 ENDIF
 IF (segmento especificado não existe)
   THEN
```

```
      .enviar mensagem de erro
      RETURN
  ENDIF
  .obter  nome  interno do segmento de usuário,  a partir do  nome
   externo   fornecido  como  parâmetro  de  entrada  e  de  dados
   armazenados na estrutura de dados do GAV
  DO  WHILE  (conjunto  de configurações  de  tela  correntes  não
              esgotado)
    DO (ET's pertencentes à CT corrente)
      IF (segmento está associado à ET)
        THEN
          IF (vai ocorrer regeneração do conteúdo da ET)
            THEN
              IF (não há superposição)
                THEN
                  .chamar,  através de função 'ESCAPE' do  GKS,  a
                   função do driver que inicializa o apontador  do
                   arquivo imagem da ET
                ELSE
                  .flag=1
                  .armazenar identificador da CT corr. em questão
                  .chamar,  através  de função 'ESCAPE' do GKS,  a
                   função  do driver que inicializa o apontador do
                   arquivo imagem da ET
              ENDIF
          ENDIF
      ENDIF
    END DO
  END DO
  .chamar a função do GKS correspondente
  IF (flag.EQ.1)
    THEN
      DO WHILE (conj. de CT correntes que necessitam regeneração)
        .apagar a tela útil
        .regenerar  conteúdo das ET's  abertas que a compõem
      END DO
  ENDIF
  RETURN



X-SET DETECTABILITY


  IF (GAV não está aberto)
    THEN
      .enviar mensagem de erro
      RETURN
  ENDIF
  IF (estado do GKS para o PV corrente é GKCL ou GKOP)
    THEN
      .enviar mensagem de erro
      RETURN
  ENDIF
  IF (nome do segmento é inválido)
```

```
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (segmento especificado não existe)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.obter nome interno do segmento de usuário, a partir do nome
 externo fornecido como parâmetro de entrada e de dados
 armazenados na estrutura de dados do GAV
.chamar a função do GKS correspondente
RETURN
```

5.8.6.2- MANIPULAÇÃO

X-CREATE SEGMENT

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente não é WSAC)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (nome do segmento é inválido)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (nome do segmento já utilizado)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.gerar nome interno para o segmento de usuário
.armazenar nomes interno e externo do segmento de usuário na
 lista de segmentos de usuário de todas as ET's ativas,
 associadas ao PV corrente, na estrutura de dados do GAV
.chamar a função do GKS 'CREATE SEGMENT'
RETURN
```

X-CLOSE SEGMENT

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente não é SGOP)
  THEN
     .enviar mensagem de erro
  ELSE
     .chamar a função do GKS 'CLOSE SEGMENT'
ENDIF
RETURN
```

X-RENAME SEGMENT

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (um dos nomes de segmento é inválido)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (novo nome do segmento já utilizado)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (segmento especificado não existe)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.substituir nome externo do segmento pelo novo nome, em todas as
 listas de segmentos de usuário presentes na tabela 4.5 da
 estrutura de dados do GAV
RETURN
```

X-DELETE SEGMENT

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (nome do segmento é inválido)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (segmento especificado não existe)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (segmento é o segmento aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.obter nome interno do segmento de usuário
DO  WHILE  (conjunto  de configurações  de  tela  correntes  não
            esgotado)
  DO  WHILE  (conjunto  de ET's pertencentes à CT  corrente  não
              esgotado)
    IF  (segmento está associado à ET e vai ocorrer  regeneração
         do conteúdo da ET)
      THEN
        IF (não há superposição)
          THEN
            .chamar, através de função 'ESCAPE' do GKS, a função
             do  driver  que inicializa o apontador  do  arquivo
             imagem da ET
          ELSE
            .flag=1
            .armazenar identificador da CT em questão
            .chamar, através de função 'ESCAPE' do GKS, a função
             do  driver  que inicializa o apontador  do  arquivo
             imagem da ET
        ENDIF
    ENDIF
  END DO
END DO
.chamar a função do GKS 'DELETE SEGMENT'
IF (flag é igual a 1)
  THEN
    DO  WHILE (conjunto de configurações de tela que  necessitam
```

```
                regeneração)
          .apagar a tela útil
          .chamar, através de função 'ESCAPE' do GAV, a função do
           driver que regenera o conteúdo das ET's abertas que a
           compõem
       END DO
 ENDIF
 .retirar nomes externo e interno do segmento em questão da
  lista de segmentos das estações de trabalho associadas ao PV
  corrente (tabela 4.5)
 RETURN


X-DELETE SEGMENT FROM WORKSTATION


 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está associada ao PV corrente)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (nome do segmento é inválido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (segmento especificado não existe na ET)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 .obter nome interno do segmento de usuário
 IF (ET pertence à CT corrente e vai ocorrer regeneração do
      conteúdo da ET)
   THEN
      IF (não há superposição)
        THEN
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que inicializa o apontador do arquivo imagem da
           ET
          .chamar a função do GKS 'DELETE SEGMENT FROM
           WORKSTATION'
        ELSE
          .apagar a tela útil
```

```
            .chamar,  através de função 'ESCAPE' do GKS, a função do
             driver  que  seta o flag de controle do  apagamento  da
             área de visualização da ET
            .chamar,  através de função 'ESCAPE' do GKS, a função do
             driver que  inicializa o apontador do arquivo imagem da
             ET
            .chamar   a   função   do   GKS   'DELETE   SEGMENT   FROM
             WORKSTATION'
            .chamar,  através de função 'ESCAPE' do GKS, a função do
             driver  que regenera o conteúdo das ET's  especificadas
             (no  caso,  as  demais ET's abertas pertencentes  à  CT
             corrente)
            .chamar,  através de função 'ESCAPE' do GKS, a função do
             driver  que reseta o flag de controle do apagamento  da
             área de visualização da ET
        ENDIF
      ELSE
        .chamar a função do GKS 'DELETE SEGMENT FROM WORKSTATION'
    ENDIF
    .retirar nomes externo e interno do segmento em questão da lista
     de segmentos da ET em questão
    RETURN


X-COPY SEGMENT TO WORKSTATION


    IF (GAV não está aberto)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (estado do GKS para o PV corrente não é WSOP ou WSAC)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (ET WISS e ET em questão não estão associadas ao PV corrente)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (nome do segmento é inválido)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (estado da ET WISS para o PV corrente é fechada)
      THEN
        .enviar mensagem de erro
        RETURN
    ENDIF
    IF (segmento especificado não existe na WISS)
      THEN
```

```
      .enviar mensagem de erro
      RETURN
ENDIF
.obter nome interno do segmento de usuário
.chamar a função do GKS 'COPY SEGMENT TO WORKSTATION'
RETURN
```

X-ASSOCIATE SEGMENT WITH WORKSTATION

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente não é WSOP ou WSAC)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET WISS e ET em questão não estão associadas ao PV corrente)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (nome do segmento é inválido)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado da ET WISS para o PV corrente é fechada)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (segmento especificado não existe na WISS)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.obter nome interno do segmento
.colocar nomes externo e interno do segmento na lista de
 segmentos da ET em questão
IF (ET pertence à CT corrente e vai ocorrer regeneração do
     conteúdo da ET)
  THEN
     IF (não há superposição)
       THEN
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que inicializa o apontador do arquivo imagem da
           ET
          .chamar a função do GKS 'ASSOCIATE SEGMENT WITH
           WORKSTATION'
```

```
        ELSE
           .apagar a tela útil
           .chamar, através de função 'ESCAPE' do GKS, a função do
            driver que seta o flag de controle do apagamento da
            área de visualização da ET
           .chamar, através de função 'ESCAPE' do GKS, a função do
            driver que inicializa o apontador do arquivo imagem da
            ET
           .chamar a função do GKS 'ASSOCIATE SEGMENT WITH
            WORKSTATION'
           .chamar, através de função 'ESCAPE' do GKS, a função do
            driver que regenera o conteúdo das ET's especificadas
            (no caso, as demais ET's abertas pertencentes à CT
            corrente)
           .chamar, através de função 'ESCAPE' do GKS, a função do
            driver que reseta o flag de controle do apagamento da
            área de visualização da ET
      ENDIF
    ELSE
      .chamar a função do GKS 'ASSOCIATE SEGMENT WITH WORKSTATION'
 ENDIF
 RETURN


X-INSERT SEGMENT


 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP ou WSOP)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (nome do segmento é inválido)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado da ET WISS para o PV corrente é fechada)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (segmento especificado não existe na WISS)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 .obter nome interno do segmento
 .chamar a função do GKS 'INSERT SEGMENT'
 RETURN
```

5.8.7- FUNÇOES DE ENTRADA

O comportamento das funções de entrada:

X-INITIALISE LOCATOR e

X-INITIALISE STROKE,

é apresentado a seguir.

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (TN inicial é inválida)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (área de echo está fora da área de visualização da ET)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
.obter número real da transformação de normalização
.chamar a função do GKS correspondente
RETURN
```

As funções de entrada:

X-INITIALISE VALUATOR,

X-INITIALISE CHOICE,

X-INITIALISE PICK e

X-INITIALISE STRING,

têm seu comportamento descrito a seguir.

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (área de echo está fora da área de visualização)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
.chamar função do GKS correspondente
RETURN
```

As funções de entrada:

X-SET LOCATOR MODE,

X-SET STROKE MODE,

X-SET VALUATOR MODE,

X-SET CHOICE MODE,

X-SET PICK MODE e

X-SET STRING MODE,

comportam-se da seguinte forma:

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
     .enviar mensagem de erro
     RETURN
```

```
 ENDIF
 .chamar função do GKS correspondente
 RETURN
```

As funções:

X-REQUEST LOCATOR e

X-REQUEST STROKE,

apresentam o comportamento abaixo.

```
 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET não está associada ao PV corrente)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (ET pertence à CT corrente)
   THEN
      .chamar função do GKS correspondente
      .obter o número virtual da transformação de normalização
   ELSE
      .enviar mensagem de erro
 ENDIF
 RETURN
```

X-REQUEST VALUATOR,

X-REQUEST CHOICE e

X-REQUEST STRING,

apresentam o comportamento abaixo.

```
 IF (GAV não está aberto)
   THEN
      .enviar mensagem de erro
      RETURN
 ENDIF
 IF (estado do GKS para o PV corrente é GKCL ou GKOP)
```

```
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de ero
    RETURN
ENDIF
IF (ET pertence à CT corrente)
  THEN
    .chamar função do GKS correspondente
  ELSE
    .enviar mensagem de erro
ENDIF
RETURN


X-REQUEST PICK


IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET pertence à CT corrente)
  THEN
    .chamar a função do GKS 'REQUEST PICK'
    IF (status é OK)
      THEN
        .obter nome externo do segmento
    ENDIF
  ELSE
    .enviar mensagem de erro
ENDIF
RETURN
```

5.8.8- FUNÇOES DE METAFILE

X-WRITE ITEM TO GKSM

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP ou WSOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
.chamar a função do GKS 'WRITE ITEM TO GKSM'
RETURN
```

As funções:

X-GET ITEM TYPE FROM GKSM e

X-READ ITEM FROM GKSM,

apresentam o comportamento mostrado a seguir.

```
IF (GAV não está aberto)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL ou GKOP)
  THEN
    .enviar mensagem de erro
    RETURN
ENDIF
IF (ET não está associada ao PV corrente)
  THEN
    .enviar mensagem de erro
  ELSE
    .chamar a função do GKS correspondente
ENDIF
RETURN
```

X-INTERPRET ITEM

```
IF (GAV não está aberto)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (estado do GKS para o PV corrente é GKCL)
  THEN
     .enviar mensagem de erro
     RETURN
ENDIF
IF (item é 'REDRAW' ou 'UPDATE')
  THEN
     IF  (alguma das ET's ativas associadas ao PV corrente,
          pertence à CT corrente e há superposição)
       THEN
          .limpar a tela útil
          DO  WHILE  (conjunto de ET's ativas pertencentes à CT
                      corrente não esgotado)
             .chamar a função do GKS 'UPDATE WORKSTATION', tendo o
              flag de regeneração o valor SUPPRESS
          END DO
          .limpar a tela útil
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que seta o flag de controle do apagamento da
           área de visualização da ET
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que inicializa o apontador do arquivo de tela
           das ET's especificadas (no caso, as ET's ativas
           pertencentes à CT corrente)
          .chamar a função do GKS 'INTERPRET ITEM'
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que reseta o flag de controle do apagamento da
           área de visualização da ET
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver que regenera o conteúdo das ET's especificadas
           (no caso as demais ET's pertencentes à CT corrente)
       ELSE
          .chamar a função do GKS 'INTERPRET ITEM'
     ENDIF
  ELSE
     IF (item é 'CLEAR WORKSTATION')
       THEN
          IF  (alguma das ET's ativas associadas ao PV corrente,
               pertence à CT corrente)
             THEN
               IF (não há superposição)
                 THEN
                    .chamar a função do GKS 'INTERPRET ITEM'
                    DO WHILE (conjunto de ET's ativas não esgotado)
                       .retraçar área de visualização
                    END DO
                 ELSE
```

```
            .limpar a tela útil
            .chamar  a  função  do  GKS  'INTERPRET ITEM'
            DO WHILE (conjunto de ET's ativas não esgotado)
             .retraçar área de visualização
            END DO
            .chamar,  através de função 'ESCAPE' do GKS,  a
             função  do driver que regenera o conteúdo  das
             ET's   especificadas  (no  caso  demais   ET's
             abertas pertencentes à CT corrente)
          ENDIF
          .chamar,  através  de  função 'ESCAPE'  do  GKS,  a
           função do driver que retira os nomes dos segmentos
           (exceto   os   segmentos   que   traçam   a   área   de
           visualização)  das  listas de segmento  das  ET's
           especificadas
        ELSE
          .chamar a função do GKS 'INTERPRET ITEM'
    ENDIF
ELSE
    IF (item é 'SET DEFERRAL STATE')
      THEN
        IF (alguma das ET's ativas associadas ao PV corrente
            pertence à CT corrente)
          THEN
            IF (não há superposição)
              THEN
                .chamar a função do GKS 'INTERPRET ITEM'
              ELSE
                IF   (deferral   mode  é  ASAP  e  Implicit
                      regeneration  mode  é  ALLOWED  e  new
                      frame  action  necessary  at  update  é
                      YES)
                   THEN
                     .limpar a tela útil
                     DO  WHILE  (conjunto  de  ET's   ativas
                                 associadas  ao  PV  corrente
                                 não esgotado)
                       .chamar   a  função  do  GKS  'UPDATE
                        WORKSTATION'
                     END DO
                     .limpar a tela útil
                     .chamar,  através de função 'ESCAPE' do
                      GKS,  a  função do driver que  seta  o
                      flag de controle do apagamento da área
                      de visualização da ET
                     .chamar,  através de função 'ESCAPE' do
                      GKS, a função do driver que inicializa
                      o  apontador do arquivo de imagem  das
                      ET's especificadas no  caso,   ET's
                      ativas  associadas  ao  PV   corrente,
                      pertencentes à CT corrente)
                      DO  WHILE  (conjunto de ET's  ativas,
                                  associadas ao PV  corrente
                                  e    pertencentes   à   CT
                                  corrente não esgotado)
```

```
                                IF  (new frame action  necessary  at
                                     update é YES)
                                  THEN
                                    .chamar    a    função    do    GKS
                                     'INTERPRET ITEM'
                                  ELSE
                                    .chamar a função do GKS  'REDRAW
                                     ALL SEGMENTS ON WORKSTATION'
                                ENDIF
                              END DO
                              .chamar, através de função 'ESCAPE' do
                               GKS,  a função do driver que regenera
                               o conteúdo das ET's especificadas (no
                               caso,     as    demais    ET's    abertas
                               pertencentes à CT corrente)
                              .chamar, através de função 'ESCAPE' do
                               GKS,  a função do driver que reseta o
                               flag  de  controle do  apagamento  da
                               área de visualização da ET
                          ELSE
                            .chamar  a  função  do  GKS  'INTERPRET
                             ITEM'
                      ENDIF
                ELSE
                  .chamar a função do GKS 'INTERPRET ITEM'
              ENDIF
            ELSE
              .chamar a função do GKS 'INTERPRET ITEM'
          ENDIF
      ENDIF
ENDIF
```

5.8.9- FUNÇOES INQUIRY

5.8.9.1- ESTADO DE OPERAÇAO

X-INQUIRE OPERATING STATE VALUE

```
IF (GAV está aberto)
  THEN
    .retorna o estado do GKS para o PV corrente,  que se encontra
    armazenado na tabela 4.4 da estrutura de dados do GAV
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

5.8.9.2- FUNÇOES INQUIRY PARA A TABELA DE DESCRIÇAO DO GKS

As funções:

X-INQUIRE LEVEL OF GKS e

X-INQUIRE LIST OF AVAILABLE WORKSTATION TYPES,

t@m seu comportamento apresentado a seguir.

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .chamar a função do GKS correspondente
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

X-INQUIRE WORKSTATION MAXIMUM NUMBERS

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .retornar o número máximo de ET's que podem estar
         simultaneamente ativas, o número máximo de ET's que
         podem estar simultaneamente abertas e o número máximo de
         ET's que podem estar associadas a um segmento, que se
         encontram armazenados na estrutura de dados do GAV
```

```
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

```
X-INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .retornar  o  número  máximo  de  TN  que  se  encontra
         armazenado na estrutura de dados do GAV
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

5.8.9.3- FUNÇOES INQUIRY PARA A LISTA DE ESTADO DO GKS

```
X-INQUIRE SET OF OPEN WORKSTATIONS

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .retornar  a  quantidade  e o conjunto  de  ET's  abertas
         associadas ao PV corrente
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

As funções:

X-INQUIRE SET OF ACTIVE WORKSTATIONS,

X-INQUIRE POLYLINE INDEX,

X-INQUIRE POLYMARKER INDEX,

X-INQUIRE TEXT INDEX,

X-INQUIRE CHARACTER HEIGHT,

X-INQUIRE CHARACTER UP VECTOR,

X-INQUIRE TEXT PATH,

X-INQUIRE TEXT ALIGNMENT,

X-INQUIRE FILL AREA INDEX,

X-INQUIRE PATTERN SIZE,

X-INQUIRE PATTERN REFERENCE POINT,

X-INQUIRE PICK IDENTIFIER,

X-INQUIRE LINETYPE,

X-INQUIRE LINEWIDTH SCALE FACTOR,

X-INQUIRE POLYLINE COLOUR INDEX,

X-INQUIRE MARKERTYPE,

X-INQUIRE MARKER SIZE SCALE FACTOR,

X-INQUIRE POLYMARKER COLOUR INDEX,

X-INQUIRE TEXT FONT AND PRECISION,

X-INQUIRE CHARACTER EXPANSION FACTOR,

X-INQUIRE CHARACTER SPACING,

X-INQUIRE TEXT COLOUR INDEX,

X-INQUIRE FILL AREA INTERIOR STYLE,

X-INQUIRE FILL AREA STYLE INDEX,

X-INQUIRE FILL AREA COLOUR INDEX e

X-INQUIRE ASPECT SOURCE FLAGS,

t@m seu comportamento descrito a seguir.

```
IF (GAV está aberto)
  THEN
     IF (estado do GKS para o PV corrente é GKCL)
       THEN
          .indicador de erro= 8
       ELSE
          .chamar a função do GKS correspondente
     ENDIF
  ELSE
```

```
    .indicador de erro=2000
ENDIF
RETURN


X-INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .retornar o número virtual da TN corrente do PV corrente,
         que se encontra armazenado na tabela 4.4 da estrutura de
         dados do GAV
     ENDIF
   ELSE
     .indicador de erro=2000
ENDIF
RETURN


X-INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .retornar uma lista contendo os números virtuais das
         transformações de normalização do PV corrente. Essa
         informação encontra-se armazenada na tabela 4.4 da
         estrutura de dados do GAV
    ENDIF
   ELSE
     .indicador de erro=2000
ENDIF
RETURN




X-INQUIRE NORMALIZATION TRANSFORMATION

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        IF (número da TN é válido)
```

```
            THEN
              .obter número real da TN
              .chamar a função do GKS 'INQUIRE NORMALIZATION
               TRANSFORMATION'
            ELSE
              .indicador de erro= 50
          ENDIF
      ENDIF
    ELSE
      .indicador de erro=2000
ENDIF
RETURN


X-INQUIRE CLIPPING INDICATOR

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente)
      THEN
        .indicador de erro= 8
      ELSE
        .chamar função do GKS 'INQUIRE CLIPPING INDICATOR'
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN


X-INQUIRE NAME OF OPEN SEGMENT

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente não é SGOP)
      THEN
        .indicador de erro= 4
      ELSE
        .chamar função do GKS 'INQUIRE NAME OF OPEN SEGMENT'
        .obter nome externo do segmento
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN

X-INQUIRE SET OF SEGMENT NAMES IN USE

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro= 7
      ELSE
        .retornar  nomes externos dos segmentos de usuário do  PV
```

```
          corrente
      ENDIF
    ELSE
      .indicador de erro=2000
ENDIF
RETURN
```

5.8.9.4- FUNÇOES INQUIRY PARA A LISTA DE ESTADO DA ESTAÇAO DE TRABALHO

As funções:

X-INQUIRE WORKSTATION CONNECTION AND TYPE,

X-INQUIRE WORKSTATION STATE,

X-INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES,

X-INQUIRE LIST OF POLYLINE INDICES,

X-INQUIRE POLYLINE REPRESENTATION,

X-INQUIRE LIST OF POLYMARKER INDICES,

X-INQUIRE POLYMARKER REPRESENTATION,

X-INQUIRE LIST OF TEXT INDICES,

X-INQUIRE TEXT REPRESENTATION,

X-INQUIRE TEXT EXTENT,

X-INQUIRE LIST OF FILL AREA INDICES,

X-INQUIRE FILL AREA REPRESENTATION,

X-INQUIRE LIST OF PATTERN INDICES,

X-INQUIRE PATTERN REPRESENTATION,

X-INQUIRE LIST OF COLOUR INDICES,

X-INQUIRE COLOUR REPRESENTATION e

X-INQUIRE WORKSTATION TRANSFORMATION,

comportam-se da seguinte maneira:

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro= 7
      ELSE
        IF (ET está associada ao PV corrente)
          THEN
            .chamar a função do GKS correspondente
          ELSE
            .indicador de erro=2004
        ENDIF
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN


X-INQUIRE SET OF SEGMENT NAMES ON WORKSTATION

IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro= 7
      ELSE
        IF (ET está associada ao PV corrente)
          THEN
            .chamar a função do GKS 'INQUIRE SET OF SEGMENT NAMES
             ON WORKSTATION'
            DO WHILE (conjunto de segmentos não esgotado)
              IF (é segmento de usuário)
                THEN
                  .obter nome externo
                  .armazenar nome externo
              ENDIF
            END DO
          ELSE
            .indicador de erro=2004
        ENDIF
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

As funções:

X-INQUIRE LOCATOR DEVICE STATE e

X-INQUIRE STROKE DEVICE STATE,

possuem o seguinte comportamento:

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro=7
      ELSE
        IF (ET está associada ao PV corrente)
          THEN
            .chamar função do GKS correspondente
            .obter número virtual da TN
          ELSE
            .indicador de erro=2004
        ENDIF
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

As funções:

X-INQUIRE VALUATOR DEVICE STATE,

X-INQUIRE CHOICE DEVICE STATE,

X-INQUIRE PICK DEVICE STATE e

X-INQUIRE STRING DEVICE STATE,

apresentam o seguinte comportamento:

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro= 7
      ELSE
        IF (ET está associada ao PV corrente)
          THEN
            .chamar a função do GKS correspondente
          ELSE
            .indicador de erro=2004
        ENDIF
    ENDIF
  ELSE
    .indicador de erro=2000
```

```
ENDIF
RETURN
```

5.8.9.5- FUNÇOES INQUIRY PARA A TABELA DE DESCRIÇAO DA ESTAÇAO DE TRABALHO

As funções:

X-INQUIRE WORKSTATION CATEGORY,

X-INQUIRE WORKSTATION CLASSIFICATION,

X-INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES,

X-INQUIRE DEFAULT DEFERRAL STATE VALUES,

X-INQUIRE POLYLINE FACILITIES,

X-INQUIRE PREDEFINED POLYLINE REPRESENTATION,

X-INQUIRE POLYMARKER FACILITIES,

X-INQUIRE PREDEFINED POLYMARKER REPRESENTATION,

X-INQUIRE TEXT FACILITIES,

X-INQUIRE PREDEFINED TEXT REPRESENTATION,

X-INQUIRE FILL AREA FACILITIES,

X-INQUIRE PREDEFINED FILL AREA REPRESENTATION,

X-INQUIRE PATTERN FACILITIES,

X-INQUIRE PREDEFINED PATTERN REPRESENTATION,

X-INQUIRE COLOUR FACILITIES,

X-INQUIRE PREDEFINED COLOUR REPRESENTATION,

X-INQUIRE LIST OF AVAILABLE GDP,

X-INQUIRE GENERALIZED DRAWING PRIMITIVE,

X-INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES,

X-INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED,

X-INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES e

X-INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES,

t&m seu comportamento descrito a seguir.

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro= 8
      ELSE
        .chamar a função do GKS correspondente
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

As funções:

X-INQUIRE DEFAULT LOCATOR DEVICE STATE,

X-INQUIRE DEFAULT STROKE DEVICE STATE,

X-INQUIRE DEFAULT VALUATOR DEVICE STATE,

X-INQUIRE DEFAULT CHOICE DEVICE STATE,

X-INQUIRE DEFAULT PICK DEVICE STATE e

X-INQUIRE DEFAULT STRING DEVICE STATE,

apresentam o comportamento abaixo.

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
      THEN
        .indicador de erro=8
      ELSE
        .obter área de echo default que se encontra armazenada na
         tabela 4.1 da estrutura de dados do GAV
        .chamar a função do GKS correspondente
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

5.8.9.6- FUNÇÕES INQUIRY PARA A LISTA DE ESTADO DO SEGMENTO

As funções:

X-INQUIRE SET OF ASSOCIATED WORKSTATIONS e

X-INQUIRE SEGMENT ATTRIBUTES,

comportam-se da seguinte maneira:

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro= 7
      ELSE
        IF (nome externo do segmento é inválido)
          THEN
            .indicador de erro= 120
          ELSE
            IF (segmento não existe para o PV corrente)
              THEN
                .indicador de erro= 122
              ELSE
                .obter nome interno do segmento
                .chamar a função do GKS correspondente
            ENDIF
        ENDIF
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

5.8.9.7- FUNÇOES INQUIRY PIXELS

As funções:

X-INQUIRE PIXEL ARRAY DIMENSIONS,

X-INQUIRE PIXEL ARRAY e

X-INQUIRE PIXEL,

têm seu comportamento descrito a seguir.

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL ou GKOP)
      THEN
        .indicador de erro= 7
      ELSE
        IF (ET está associada ao PV corrente)
          THEN
            .chamar a função do GKS correspondente
          ELSE
            .indicador de erro=2004
        ENDIF
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

5.8.10- FUNÇÕES UTILITÁRIAS

As funções:

X-EVALUATE TRANSFORMATION MATRIX e

X-ACCUMULATE TRANSFORMATION MATRIX,

comportam-se da forma descrita a seguir.

```
IF (GAV está aberto)
  THEN
    IF (estado do GKS para o PV corrente é GKCL)
       THEN
         .enviar mensagem de erro
       ELSE
         .chamar a função do GKS correspondente
    ENDIF
ENDIF
RETURN
```

5.8.11- FUNÇOES PARA TRATAMENTO DE ERROS

X-EMERGENCY CLOSE GKS

```
IF (estado do GKS para o PV corrente é SGOP)
  THEN
    .chamar a função do GKS 'CLOSE SEGMENT'
    .atualizar  o estado do GKS para o PV corrente na tabela  4.4
     da estrutura de dados do GAV
ENDIF
IF (estado do GKS para o PV corrente é WSAC)
  THEN
    DO  WHILE (conjunto de ET's ativas associadas ao PV  corrente
               não esgotado)
      .chamar a função do GKS 'DEACTIVATE WORKSTATION'
    END DO
    .atualizar  o estado do GKS para o PV corrente para WSOP,  na
     tabela 4.4 da estrutura de dados do GAV
ENDIF
IF (estado do GKS para o PV corrente é WSOP)
  THEN
    DO  WHILE (conjunto de ET's abertas associadas ao PV corrente
               não esgotado)
      IF (ET não é WISS)
       THEN
         .retirar  identificador da ET da tabela 4.5 da estrutura
          de dados do GAV
       ELSE
         .atualizar  entrada 'estado da ET WISS para o  PV'  para
          'FECHADA'
      ENDIF
      IF (ET pertence à categoria OUTIN)
        THEN
          .chamar, através de função 'ESCAPE' do GKS, a função do
           driver  que retira identificador de conexão da  ET  de
           sua estrutura de dados
      ENDIF
    END DO
    IF  (alguma  das  ET'S  abertas  associadas  ao  PV  corrente
         pertence A CT corrente)
      THEN
        IF (há superposição)
          THEN
            .limpar a tela útil
            .chamar, através de função 'ESCAPE' do GKS, a função
             do driver que seta o flag de controle do  apagamento
             da área de visualização da ET
            DO  WHILE (conjunto de ET's abertas associadas ao  PV
                       corrente)
              .chamar a função do GKS 'CLOSE WORKSTATION'
            END DO
            .chamar,  através de função 'ESCAPE' do GKS, a função
             do   driver   que  regenera  o  conteúdo  das   ET's
```

```
                especificadas (no caso,  ET's abertas pertencentes à
                CT corrente)
               .chamar,  através de função 'ESCAPE' do GKS, a função
                do  driver  que  reseta  o  flag  de  controle  do
                apagamento da área de visualização da ET
            ELSE
              DO  WHILE (conjunto de ET's abertas associadas ao  PV
                         corrente não esgotado)
                .chamar função do GKS 'CLOSE WORKSTATION'
              END DO
          ENDIF
        ELSE
          DO  WHILE  (conjunto  de ET's abertas  associadas  ao  PV
                      corrente não esgotado)
            .chamar a função do GKS 'CLOSE WORKSTATION'
          END DO
      ENDIF
      .atualizar o estado do GKS para o PV corrente para  GKOP,  na
       tabela 4.4 da estrutura de dados do GAV
ENDIF
IF (estado do GKS para o PV corrente é GKOP)
  THEN
    IF (estado do GKS para os demais PV é GKCL)
      THEN
        .chamar a função do GKS 'CLOSE GKS'
    ENDIF
    .atualizar  o estado do GKS para o PV corrente para GKCL,  na
     tabela 4.4 da estrutura de dados do GAV
ENDIF
RETURN


X-ERROR LOGGING

 .chamar a função do GKS 'ERROR LOGGING'
 RETURN


X-ERROR HANDLING

 .setar flag de erro
 .colocar  no arquivo geral de erros o identificador do  erro,  o
  identificador  da rotina que gerou o erro,  o identificador  do
  processo visual onde o erro foi gerado e uma mensagem de erro
 IF (estado do GKS para o PV corrente não é GKCL)
   THEN
     .obter nome do arquivo de erros do PV corrente
     .chamar a função do GKS 'ERROR LOGGING'
 ENDIF
 RETURN
```

5.9- ACESSO A INFORMAÇOES DA ESTRUTURA DE DADOS DO GAV

O GAV deve ainda oferecer funções que permitam ao usuário acessar informações contidas em sua estrutura de dados. São as seguintes as funções oferecidas:

1- inquire conjunto de estações de trabalho definidas;

2- inquire conjunto de processos visuais definidos;

3- inquire conjunto de configurações de tela definidas;

4- inquire estação de trabalho;

5- inquire processo visual;

6- inquire configuração de tela;

7- inquire processo visual corrente;

8- inquire configuração de tela corrente;

9- inquire máximos.

O comportamento dessas funções será descrito a seguir.

INQUIRE CONJUNTO DE ESTAÇOES DE TRABALHO DEFINIDAS

```
IF (GAV está aberto)
  THEN
    .retornar o identificador e a categoria das estações de
     trabalho definidas
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

INQUIRE CONJUNTO DE PROCESSOS VISUAIS DEFINIDOS

```
IF (GAV está aberto)
  THEN
    .retornar identificadores dos processos visuais definidos
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

```
INQUIRE CONJUNTO DE CONFIGURAÇOES DE TELA DEFINIDAS

IF (GAV ESTA ABERTO)
  THEN
    .retornar identificadores das configurações de tela definidas
  ELSE
    .indicador de erro=2000
ENDIF
RETURN


INQUIRE ESTAÇAO DE TRABALHO


IF (GAV está aberto)
  THEN
    IF (ET foi definida)
      THEN
        IF (identificadorda ET é válido)
          THEN
            IF (ET é OUTIN)
              THEN
                .retornar identificador de conexão, tipo, área de
                 visualização, prioridade e índice de corr. da ET
              ELSE
                .retornar   identificador   de   conexão,   tipo   e
                 categoria da ET em questão
            ENDIF
          ELSE
            .indicador de erro=20
        ENDIF
      ELSE
        .indicador de erro=2001
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN



INQUIRE PROCESSO VISUAL

IF (GAV está aberto)
  THEN
    IF (identificador do PV é válido)
      THEN
        IF (processo visual foi definido)
          THEN
            .retornar  identificadores das estações  de  trabalho
             que estão associadas ao PV em questão
          ELSE
            .indicador de erro=2003
        ENDIF
      ELSE
```

```
        .indicador de erro=2013
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN


INQUIRE CONFIGURAÇÃO DE TELA


IF (GAV está aberto)
  THEN
    IF (identificador da CT é válido)
      THEN
        IF (configuração de tela foi definida)
          THEN
            .retornar identificadores e tipo das estações de
             trabalho que compõem a CT em questão
          ELSE
            .indicador de erro=2002
        ENDIF
      ELSE
        .indicador de erro=2016
    ENDIF
  ELSE
    .indicador de erro=2000
ENDIF
RETURN



INQUIRE PROCESSO VISUAL CORRENTE

IF (GAV está aberto)
  THEN
    .retornar o identificador do processo visual corrente
  ELSE
    .indicador de erro=2000
ENDIF
RETURN



INQUIRE CONFIGURAÇÃO DE TELA CORRENTE

IF (GAV está aberto)
  THEN
    IF (tipo é válido)
      THEN
        .retornar o identificador da configuração de tela
         corrente
      ELSE
        .indicador de erro=22
  ELSE
```

```
    .indicador de erro=2000
ENDIF
RETURN


INQUIRE MAXIMOS

IF (GAV está aberto)
  THEN
    .retornar número máximo de ET's OUTIN, de ET's pertencentes
     às demais categorias, de processos visuais e de
     configurações de tela que podem ser definidas
  ELSE
    .indicador de erro=2000
ENDIF
RETURN
```

Como já se pôde perceber neste capítulo, o GAV exige a presença de uma interface especial GKS-DISPOSITIVO. Esta interface será o assunto abordado pelo próximo capítulo.

6- INTERFACE ESPECIAL GKS-DISPOSITIVO

A inclusão de facilidades de gerenciamento de janelas em um sistema gráfico interativo que utiliza um pacote gráfico GKS, exige por um lado a definição de um elemento, o GAV, que se sobrepõe ao GKS no controle das diversas áreas de visualização. Por outro lado, a inclusão dessas facilidades exige a definição de uma interface especial entre o GKS e o dispositivo gráfico; interface esta, que será discutida neste capítulo.

A figura 6.1 mostra o modelo de um sistema gráfico interativo com GAV.

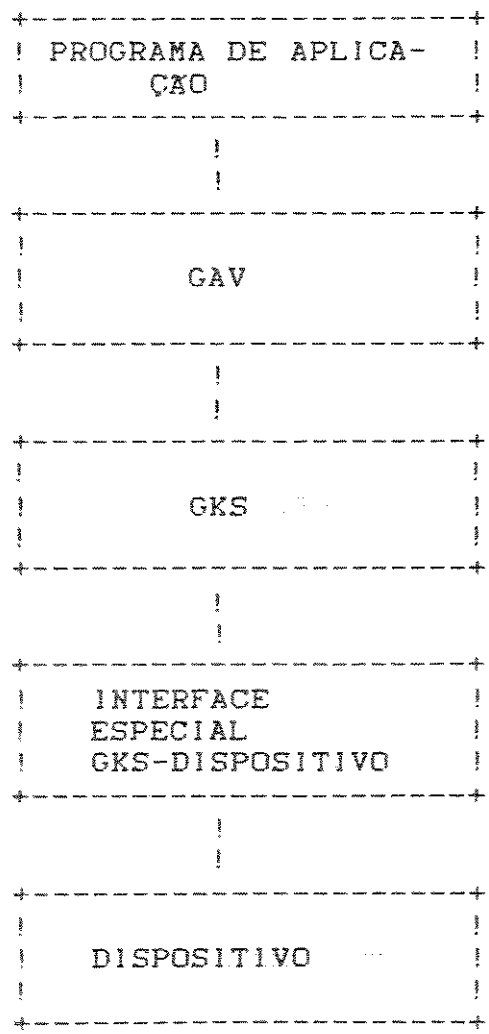

figura 6.1- Modelo de um sistema gráfico interativo com GAV

Uma vez que a cada estação de trabalho OUTIN definida está associada uma fração retangular da superfície total de visualização oferecida pelo dispositivo físico, a interface GKS-DISPOSITIVO deve ser capaz de endereçar funções apenas a uma dessas áreas de visualização. Por exemplo, ele deve oferecer uma função para limpar uma fração da superfície total de visualização, oferecida pelo dispositivo. Esses recursos não são previstos nas interfaces GKS-DISPOSITIVO normalmente oferecidas.

Além disso, a interface especial GKS-DISPOSITIVO deve manter um arquivo de tela para cada estação de trabalho aberta, contendo as primitivas de saída geradas. Esses arquivos são utilizados quando uma regeneração de tela se faz necessária.

Para poder oferecer todos esses recursos, a interface especial GKS-DISPOSITIVO deve dispor de uma estrutura de dados própria. Essa estrutura de dados pode ser vista na figura abaixo.

- TABELA: ET's OUTIN abertas

ident de conexão	área de visual.	priori-dade	índice de cor	lista de seg.	visibi-lidade	detec-tabili-dade	ret. de seg.	ident. de pick	retân-gulo de pick

- Conjunto de ET's que compõem a CT corrente
- Flag que controla o apagamento da área de visualização da ET
- Flag que indica se existe segmento aberto
- Identificador de pick corrente
- Arquivos sequenciais contendo as primitivas de saída das ET's (um para cada ET OUTIN aberta)

figura 6.2- Estrutura de dados da interface especial GKS-DISPOSITIVO

As funções que compõem a interface especial GKS-DISPOSITIVO podem ser classificadas em três grupos:

1- funções presentes na interface normal GKS-DISPOSITIVO, cujo comportamento foi modificado. Esse é o caso, por exemplo da função 'clear screen', que na interface normal GKS-DISPOSITIVO limpa a tela útil e na interface especial GKS-DISPOSITIVO limpa apenas a área de visualização da ET em questão;

2- funções específicas da interface especial GKS-DISPOSITIVO, chamadas através de função 'ESCAPE' do GKS. Constitue um bom exemplo deste grupo, a função dessa interface que regenera o conteúdo das ET's fornecidas como parâmetro de entrada;

3- funções presentes na interface normal GKS-DISPOSITIVO e cujo comportamento não foi modificado.

A seguir, é fornecido o comportamento das funções da interface especial GKS-DISPOSITIVO.

6.1- GRUPO 1

OPEN DEVICE

```
IF (ET pertence à CT corrente)
  THEN
    IF (flag de controle do apagamento está OFF)
       THEN
         .limpar a área de visualização da ET
    ENDIF
    .armazenar contorno da área de visualização da ET no arquivo
     de tela
    .gerar contorno da área de visualização da ET
ENDIF
.abrir arquivo de tela da ET
RETURN
```

```
CLOSE DEVICE

 IF (ET pertence à CT corrente)
   THEN
     IF (flag de controle do apagamento está OFF)
        THEN
          .limpar a área de visualização da ET
     ENDIF
 ENDIF
 .fechar arquivo de tela da ET
 RETURN


CLEAR SCREEN


 IF (ET pertence à CT corrente)
   THEN
     IF (flag de controle do apagamento está OFF)
       THEN
         .limpar área de visualização da ET
     ENDIF
     .gerar contorno da área de visualização da ET
 ENDIF
 RETURN


CLEAR WORKSTATION


 IF (ET pertence à CT corrente)
   THEN
     IF (flag de controle do apagamento está OFF)
       THEN
         .limpar a área de visualização da ET
     ENDIF
     .gerar contorno da área de visualização da ET
 ENDIF
 .zerar lista de segmentos da ET
 RETURN


FUNÇOES DE SAIDA


 IF (ET pertence à CT corrente)
   THEN
     IF (transformação de segmento e transformação de workstation
         ainda não foram realizadas)
       THEN
         .realizar transformação de segmento
         .realizar transformação de workstation
     ENDIF
     IF (existe segmento aberto)
```

```
        THEN
          .armazenar/atualizar retângulos de segmento e de pick
      ENDIF
      .atualizar identificador de pick corrente, se necessário
      .armazenar saída (tipo de primitiva, dados, atributos e
       retângulo de clipping) no arquivo de tela da ET em questão
      .realizar o clipping da primitiva de saída
      .executar a saída
 ENDIF
 RETURN


REQUEST LOCATOR


 .chamar a função REQUEST LOCATOR da interface normal GKS-
  DISPOSITIVO
 IF (ponto obtido está fora da área de visualização da ET)
   THEN
     .status= NONE
 ENDIF
 RETURN



REQUEST STROKE


 .chamar a função REQUEST STROKE da interface normal GKS-
  DISPOSITIVO
 IF (algum dos pontos obtidos está fora da área de visualização
      da ET)
    THEN
     .status= NONE
 ENDIF
 RETURN


REQUEST PICK


 .chamar a função REQUEST LOCATOR da interface normal GKS-
  DISPOSITIVO
 .obter nome do segmento e identificador de pick, utilizando os
  retângulos de segmento e de pick armazenados na tabela
 IF (segmento não consta na lista de segmentos da ET em questão)
   THEN
     .status= NOPICK
 ENDIF
 RETURN
```

CREATE SEGMENT

.armazenar nome do segmento na lista de segmentos da ET em questão
.atualizar flag que indica se existe segmento aberto para o valor SIM
RETURN

CLOSE SEGMENT

.atualizar flag que indica se existe segmento aberto para o valor NÃO
RETURN

DELETE SEGMENT

.retirar nome do segmento da lista de segmentos da ET especificada
RETURN

SET VISIBILITY

.atualizar a visibilidade do segmento na tabela
RETURN

SET DETECTABILITY

.atualizar a detectabilidade do segmento na tabela
RETURN

6.2- GRUPO 2

APAGAR A TELA ÚTIL

.limpar a superfície de visualização oferecida pelo dispositivo físico
RETURN

REGENERAR CONFIGURAÇÃO DE TELA

```
DO WHILE (conjunto de ET's fornecidas não esgotado)
   DO WHILE (arquivo não finalizado)
      .ler registro do arquivo de tela da ET
      .realizar clipping
      .executar saída
   END DO
END DO
RETURN
```

ATUALIZAR VALOR DO FLAG DE APAGAMENTO

.atualizar o flag de controle do apagamento da área de visualização da ET
RETURN

INICIALIZAR APONTADOR DO ARQUIVO DE TELA

.inicializa o apontador do arquivo de tela da ET especificada
.armazenar contorno da área de visualização da ET
RETURN

ATUALIZAR CONFIGURAÇÃO DE TELA CORRENTE

.atualiz o conjunto de ET's que compõem a CT corrente
RETURN

GERAR CONTORNO

.gerar o contorno da área de visualização da ET especificada
RETURN

ARMAZENAR IDENTIFICADOR DE CONEXÃO, AREA DE VISUALIZAÇÃO e PRIORIDADE

.armazenar identificador de conexão, área de visualização, prioridade e índice de cor da ET na tabela da estrutura de dados da interface especial GKS-DISPOSITIVO
RETURN

RETIRAR IDENTIFICADOR DE CONEXÃO

.retirar o identificador de conexão da ET da tabela da estrutura de dados da interface especial GKS-DISPOSITIVO
RETURN

ATUALIZAR AREA DE VISUALIZAÇÃO

.atualizar a área de visualização da ET em questão na tabela da estrutura de dados da interface especial GKS-DISPOSITIVO
RETURN

ATUALIZAR PRIORIDADE

.atualizar a prioridade da ET em questão na tabela da estrutura de dados da interface especial GKS-DISPOSITIVO
RETURN

INICIALIZAR DISPOSITIVO

.inicializar o dispositivo físico
RETURN

FINALIZAR DISPOSITIVO

.resetar o dispositivo físico
RETURN

6.3- GRUPO 3

As demais funções, existentes em uma interface comum GKS-DISPOSITIVO compõem este grupo. Não são fornecidos, no presente trabalho, os comportamentos destas funções, uma vez que as mesmas não são afetadas pela presença do GAV.

O próximo capítulo apresenta uma análise do uso de um pacote gráfico GKS como suporte ao GAV.

7- ANALISE DO USO DE UM PACOTE GRAFICO GKS COMO SUPORTE AO GAV

No capítulo 4 deste trabalho, viu-se que a solução adotada para o problema de projeto e implementação do Gerenciador de Areas de Visualização (GAV), foi a de utilizar um pacote gráfico GKS como suporte ao referido sistema. Certamente esta solução não é única. Poder-se-ia ter optado, por exemplo, pela implementação do GAV diretamente sobre os recursos oferecidos pelo dispositivo.

A principal vantagem, advinda do uso de um pacote gráfico GKS como suporte ao GAV, é a alta transportabilidade alcançada no mesmo, uma vez que o GKS é a norma internacional, estabelecida pela ISO (International Standards Organization) para a implementação de pacotes gráficos (ISO-IS7942).

Pode-se citar como desvantagem, o fato de que o GAV, projetado e implementado desta forma, oferece para muitas aplicações, muito mais recursos do que aqueles por elas exigidos, tornando-se um sistema oneroso (em termos de memória e tempo de processamento) para tais situações.

Outras desvantagens apresentam-se, quando considera-se o GAV como sendo um dos componentes de um SGIU. Estas desvantagens resultam das inadequações apresentadas pelos pacotes gráficos GKS quando utilizados como suporte a SGIU /GIIT-83/.

A principal inadequação deve-se ao modelo de entrada/saída apresentado pela norma GKS. Este modelo separa entrada e echo léxico de saída, ficando a cargo do implementador do sistema a determinação dos tipos de echo disponíveis.

No entanto, a geração de echos é uma atribuição

importante da interface de usuário e como tal deve ser controlada pelo SGIU, o que não é possível com a utilização de um pacote gráfico GKS.

Uma outra inadequação dos pacotes gráficos GKS quando considerados como suporte a SGIU, está relacionada com o conceito de dispositivos lógicos em que se baseia a norma GKS.

Segundo este conceito, ao programador não é dada a capacidade de distinguir dispositivos físicos diferentes que retornem o mesmo tipo de dado; ou seja, o programador é isolado dos detalhes comportamentais dos dispositivos. Ocorre, no entanto, que esses detalhes estão intimamente relacionados com a qualidade da interface de usuário, ponto cruxial da aplicação de SGIU.

Apesar dessas desvantagens, a alta transportabilidade obtida no GAV com a utilização dos recursos de um pacote gráfico GKS, pesou o suficiente para que esta solução fosse adotada. Deve-se acrescentar que em inúmeros casos estes fatores não são importantes.

O próximo capítulo apresenta um exemplo de utilização do GAV. Trata-se de uma ferramenta para apoio à definição e manipulação de banco de dados.

8- EXEMPLO DE UTILIZAÇÃO DO GAV

O exemplo de utilização do GAV, apresentado neste capítulo, consiste na implementação parcial de uma ferramenta para definição e manipulação de bases de dados. Essa ferramenta, na forma aqui apresentada, é produto dos primeiros esforços realizados na concepção da LIGG (Linguagem Gráfica Baseada em Grafo para o MER/PAC) /QUE-86/.

O próximo item fornece uma descrição passo a passo desse sistema e o apêndice B contém o resultado de sua implementação.

8.1- DESCRIÇÃO DO PROBLEMA

PASSO 1: nesse passo o menu de entrada (fig. 8.1) é apresentado ao usuário, sendo então, solicitada ao mesmo a escolha de uma opção. Se a opção escolhida for a quarta, o usuário deseja sair do sistema e o programa conclui a sua execução. Se a opção escolhida for a primeira, o passo 2 será o próximo passo a ser executado. A segunda opção permite a manipulação de uma base de dados anteriormente definida e a terceira, fornece informações sobre o comportamento do sistema ao usuário. Essas duas opções não foram implementadas.

PASSO 2: o programa obtém do usuário as seguintes informações (fig.8.2):

-nome da base de dados,

-versão,

-autor,

-data e segue para o passo 3.

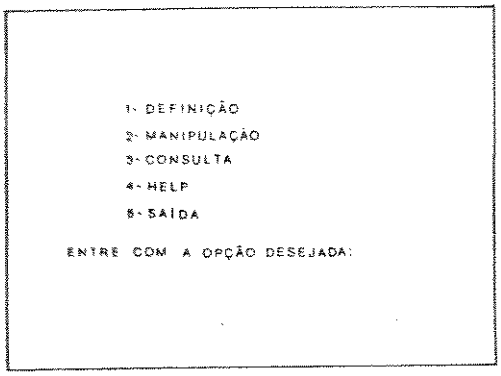

fig. 8.1

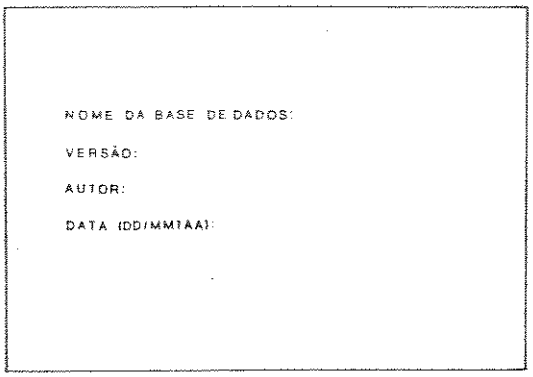

fig.8.2

PASSO 3: a CT da fig.8.3 é apresentada ao usuário. A área 1 é a área de trabalho que será utilizada pelo usuário para o desenho de esquemas e manipulação da base de dados. A área 2 apresenta o menu de tipos de primitivas disponíveis (entidade, relacionamento, esquema e agrupamento). A área 3 informa ao usuário o nome da base de dados em criação e solicita ao mesmo a escolha de uma das opções apresentadas na área 4. Se a opção escolhida for a segunda, o passo 1 é o próximo passo a ser executado e o menu de entrada é novamente apresentado ao usuário. Caso contrário, ou seja, caso a escolha seja a opção 1, o passo 4 é executado.

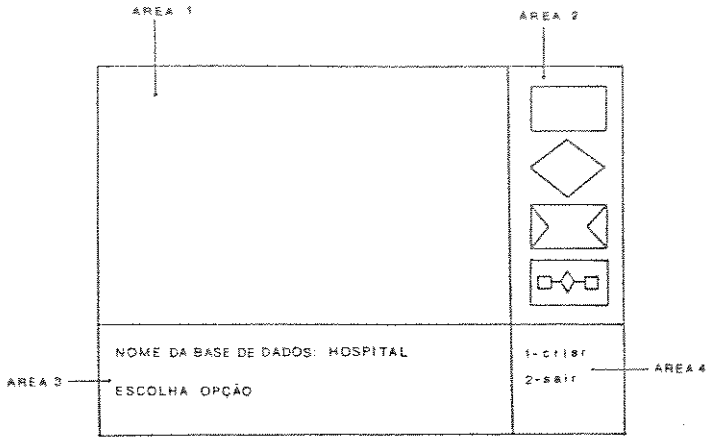

fig.8.3

PASSO 4: nesse passo o programa solicita ao usuário que ele escolha um dos tipos de primitiva e segue para o passo seguinte (fig. 8.4).

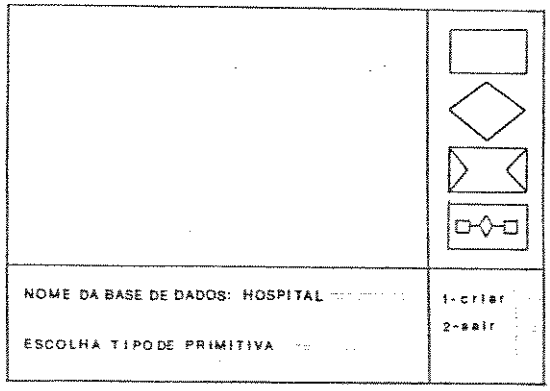

fig.8.4

PASSO 5: as áreas 2 e 3 têm seus conteúdos modificados (figuras 8.5, 8.6, 8.7 ou 8.8, dependendo se o tipo de primitiva escolhido for respectivamente: entidade, relacionamento, esquema ou agrupamento). E então, realizada uma solicitação ao usuário para que ele

escolha uma das opções do menu apresentado na área 4. Caso a opção 6 seja a opção escolhida, o passo 3 é o próximo passo a ser executado. Caso o usuário escolha a opção 1, o passo seguinte será executado. Caso a opção 4 seja a escolhida, o próximo passo a ser executado é ov passo 10. As demais opções permitem a remoção ou alteração de uma primitiva ou ainda, a listagem de todas as informações a ela associadas. Estas opções não foram implementadas.

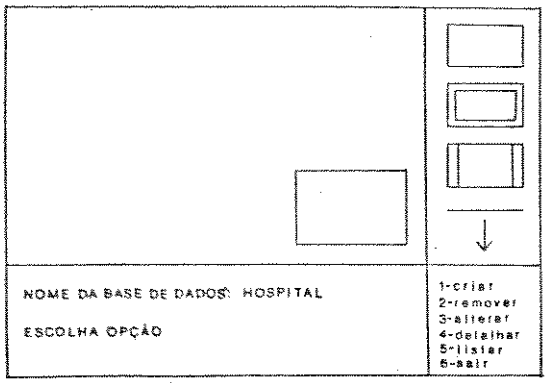

fig.8.5

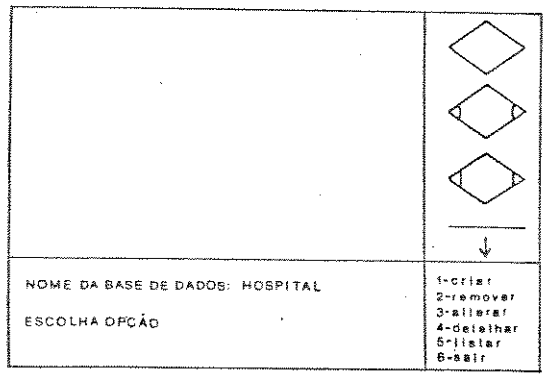

fig.8.6

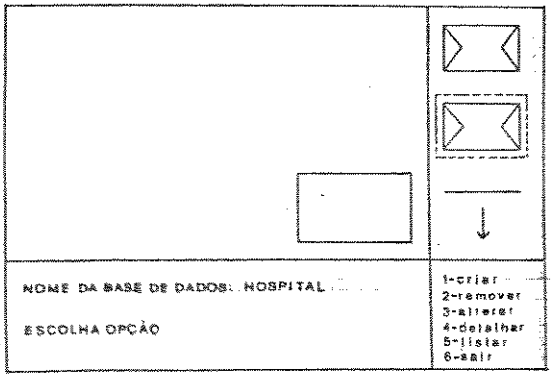

fig.8.7

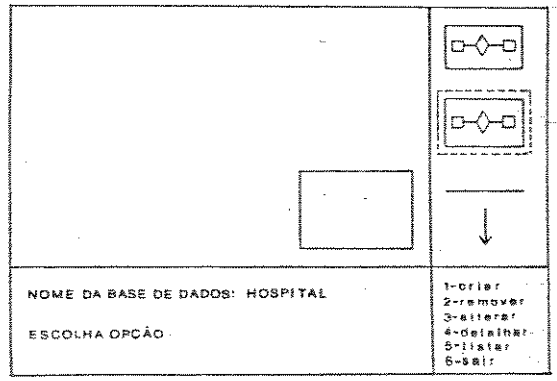

fig.8.8

PASSO 6: nesse passo é realizada uma solicitação ao usuário para que ele escolha a primitiva desejada dentro do tipo escolhido previamente (fig. 8.9, supondo que entidade foi o tipo de primitiva escolhido pelo usuário no passo anterior).

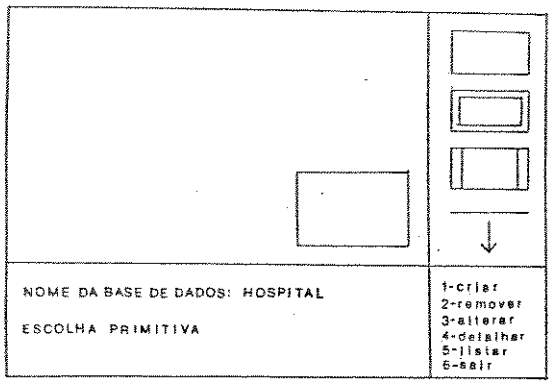

fig. 8.9

PASSO 7: uma vez escolhida a primitiva desejada, o programa solicita ao usuário que ele posicione o cursor na localização desejada da área 1 (fig. 8.10) e segue para o passo seguinte.

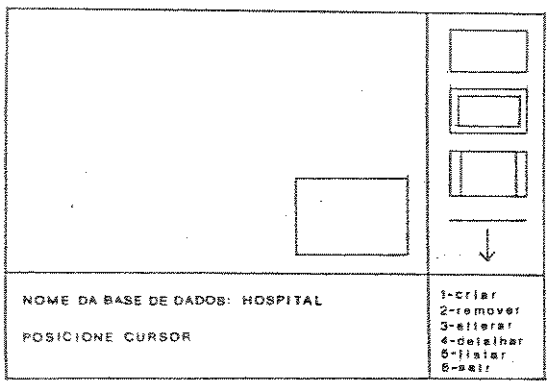

fig. 8.10

PASSO 8: a primitiva selecionada é desenhada a partir da posição fornecida e o programa solicita ao usuário que ele insira o texto referente à primitiva escolhida. É importante observar que as primitivas ligação e ligação com seta não possuem textos associados (fig. 8.11, supondo que o usuário escolheu criar uma entidade regular).

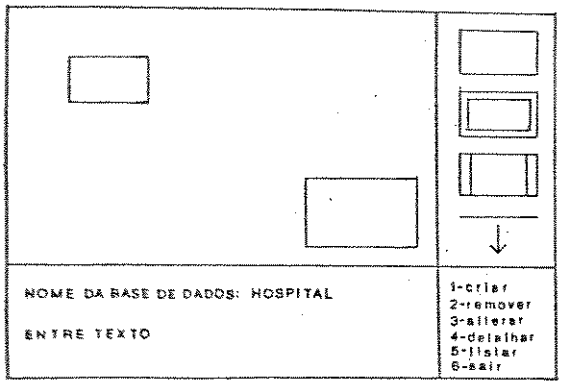

fig. 8.11

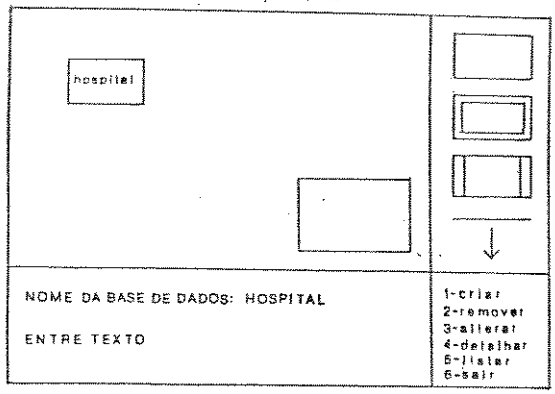

fig. 8.12

PASSO 9: neste passo, o programa solicita ao usuário que ele posicione o cursor no ponto a partir do qual ele deseja que o texto seja escrito, gera o texto e volta para o passo 5 (fig. 8.12, supondo que o texto inserido foi "HOSPITAL").

PASSO 10: o programa solicita ao usuário que escolha o detalhe e apresenta-o na área 5.

8.2- UTILIZAÇÃO DOS RECURSOS DO GAV

Para resolver o problema descrito no item 8.1, foi necessário definir:

1- Seis estações de trabalho pertencentes à categoria OUTIN. Uma cuja área representa toda a tela oferecida pelo dispositivo (ET1) e as demais com as áreas apresentadas na fig. 8.3.

2- Um processo visual ao qual estão associadas as estações de trabalho mencionadas no sub-item anterior, além da estação de trabalho pertencente à categoria WISS.

3- Duas configurações de tela. A primeira composta apenas pela ET1 e a segunda composta pelas outras cinco estações de trabalho OUTIN referidas no sub-item 1.

A apresentação dos diferentes menus em cada uma das áreas foi controlada através da utilização das funções: X-SET CT CORRENTE, X-SET VISIBILITY e X-REDRAW ALL SEGMENTS ON WORKSTATION.

O resultado da implementação pode ser visto no apêndice B.

No capítulo seguinte, são colocadas as considerações finais sobre o presente trabalho.

9- CONSIDERAÇÕES FINAIS

O Gerenciador de Recursos Gráficos, Gerenciador de Áreas de Visualização (GAV), proposto neste trabalho, consiste numa ferramenta que possibilita a inclusão de facilidades de gerenciamento de janelas em um sistema gráfico interativo.

Ele utiliza como suporte os recursos oferecidos por um pacote gráfico GKS e foi definido tendo-se em vista a sua utilização em SGIU.

Sendo asssim, pode-se afirmar que o presente trabalho apóia-se em estudos realizados, principalmente, nas áreas de interação homem-máquina e padronização gráfica.

A apresentação de uma metodologia de desenvolvimento do GAV, com o fornecimento de sua especificação funcional e de seu projeto de implementação, consiste numa valorosa contribuição para outras implementações desse sistema.

Vários são os recursos oferecidos pelo GAV aos projetistas dos programas de aplicação. Esses recursos são colocados à disposição desses projetistas sob a forma de uma biblioteca de funções, as quais podem ser classificadas em três grandes grupos:

1- Funções de controle: relacionadas com inicialização da estrutura de dados do GAV, tratamento de erros e definições. São elas:
   - OPEN GAV,
   - CLOSE GAV,
   - DEFINE ET,
   - DEFINE PV,

- DEFINE CT,
- SET PV CORRENTE,
- SET CT CORRENTE e
- SET PRIORIDADE.

2- Funções inquiry para a estrutura de dados do GAV: acessam as informações contidas na estrutura de dados do GAV. São as seguintes:
- INQUIRE ET DEFINIDAS,
- INQUIRE PV DEFINIDOS,
- INQUIRE CT DEFINIDAS,
- INQUIRE ET,
- INQUIRE PV,
- INQUIRE CT,
- INQUIRE PV CORRENTE,
- INQUIRE CT CORRENTE e
- INQUIRE MAXIMOS.

3- Funções que permitem a utilização dos recursos GKS em cada PV: uma função do GAV para cada função oferecida pelo GKS, denominadas da mesma forma que as funções do GKS, acrescidas de um X à frente.

A inclusão de facilidades de gerenciamento de janelas em um sistema gráfico interativo que utiliza um pacote gráfico GKS, exige por um lado a definição de um elemento, o GAV, que se sobrepõe ao GKS no controle das diversas áreas de visualização. Por outro lado, a inclusão dessas facilidades exige a definição de uma interface especial, mais complexa, entre o GKS e o dispositivo gráfico (fig. 9.1). Essa interface pode ser vista

como sendo composta por duas camadas. Uma, mais próxima do dispositivo gráfico, que consiste na interface normal GKS-DISPOSITIVO e outra, situada acima da primeira, que reúne as características especiais exigidas pela inclusão de facilidades de gerenciamento de janelas. Essas características especiais são acessadas através de funções 'ESCAPE' do GKS.

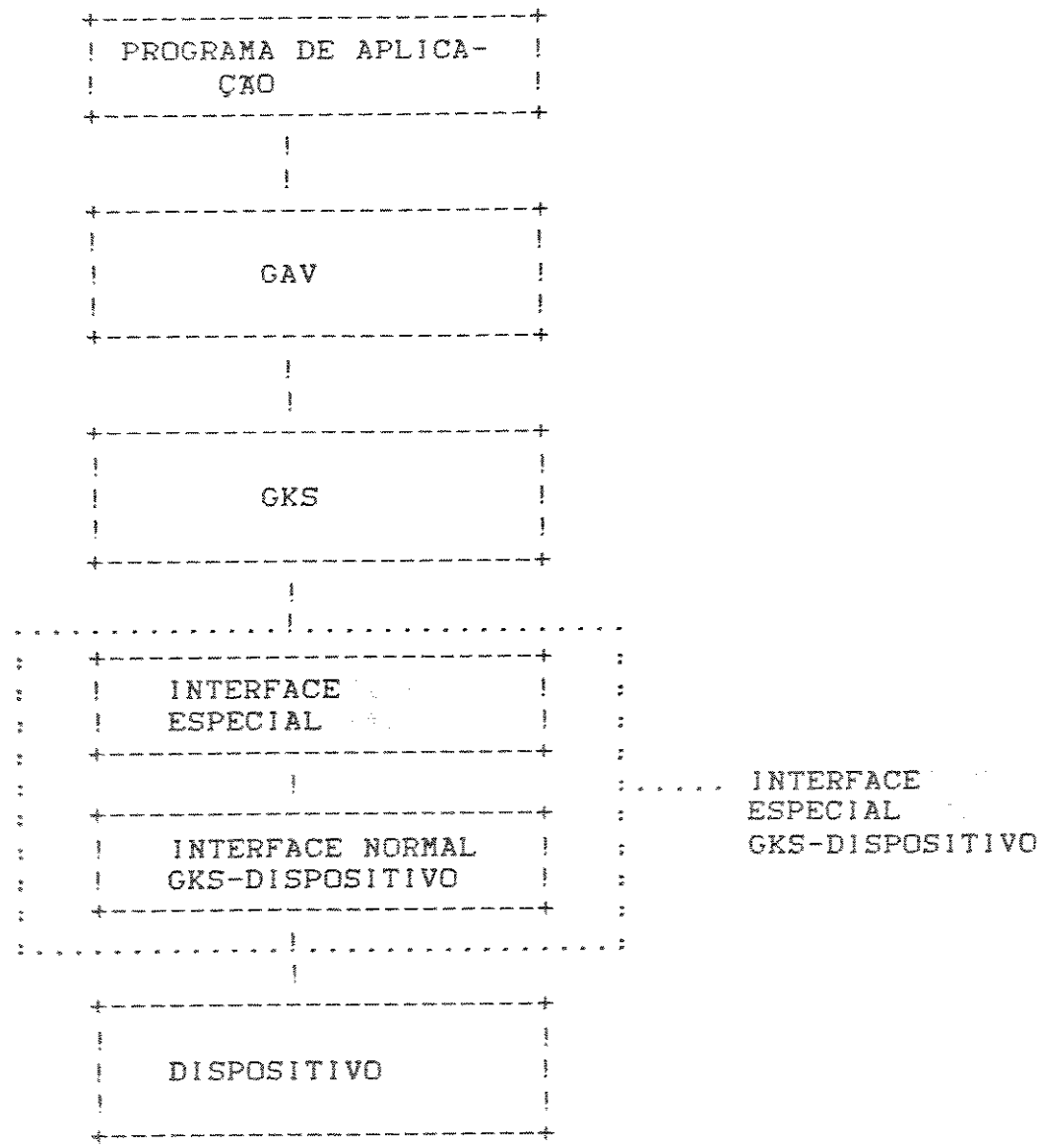

figura 9.1- Modelo de um sistema gráfico interativo com GAV

É importante frisar que a solução adotada neste trabalho, para o problema de projeto e implementação de um Gerenciador de Recursos Gráficos, certamente não é única. Poder-se-ia ter optado, por exemplo, pela implementação do GAV diretamente sobre os recursos oferecidos pelo dispositivo, em vez de utilizar os recursos oferecidos por um pacote gráfico GKS.

O capítulo 7 enfocou vantagens e desvantagens dessa solução, conforme a seguir:

- A principal vantagem decorrente do uso de um pacote gráfico GKS como suporte ao GAV é a alta transportabilidade alcançada no mesmo.
- Como desvantagens pode-se citar:
    - oferecimento, para muitas aplicações, de muito mais recursos além daqueles por elas exigidos, o que torna a sua utilização onerosa para tais situações;
    - desvantagens, resultantes das inadequações apresentadas pelos pacotes gráficos GKS quando utilizados como suporte a SGIU, relacionadas com: modelo ENTRADA/SAÍDA apresentado pela norma GKS e conceito de dispositivos lógicos de entrada em que se baseia essa norma.

Apesar dessas desvantagens, a alta transportabilidade obtida no GAV com a utilização dos recursos de um pacote gráfico GKS, pesou o suficiente, neste caso, para que esta solução fosse adotada.

Finalmente, seguindo esta mesma linha de pesquisa, são sugeridos como futuros trabalhos, a especificação, o projeto e a implementação dos demais componentes do Processador de Diálogos.

10- BIBLIOGRAFIA

10.1- REFERENCIAS

/ENC-79/ - Encarnação,J.L.- "Interactive Computer Graphics- the rich and dynamic man-machine environment", EURO IFIP- North Holland Publishing Company, 1979, pp 521/529

/GIIT-83/ - GRAPHICAL INPUT INTERACTION TECHNIQUE WORKSHOP SUMMARY- Computer Graphics, jan-1983

/HAN-84/ - Hanusa, H.; Kuhlmann, H.W.; Pfaff, G.E.- "On Constructing Interactive Graphics Systems", Technische Hochschule Darmstadt

/HEL-86/ - Hartelt, Helga I. M. - "Implementação do GAV e da Interface Especial GKS-Dispositivo"- Relatório Interno em fase de elaboração - UNICAMP

/MAG-85/ - Magalhães, L.P.; Tozzi, C.L.- "Análise Inicial da Interface entre a Aplicação e o Diálogo"- Relatório Interno CTI/IA, maio-1985

/OLSEN-84/- Olsen, D.R.; Buxton, W.; Ehrich, R.; Kasik, D.J.; Rhyne, J.R.; Sibert, J.- "A Context for User Interface Management"- IEEE Computer Graphics and Applications, dezembro-1984

/PFA-83/ - Pfaff, G.E.- "The Consruction of Operator Interfaces Based on Logical Input Devices" - Acta Informatica 19, 1983, pp 151/166

/QUE-86/ - Quezada, G.,S. - "Implementação de uma Ferramenta Gráfica para o Modelo Entidade Relacionamento - PAC", Relatório Interno do grupo PAC/BD - FEE/UNICAMP, junho 1986

/SIL-85/ - Silva, M.V. - "Processador de Diálogo: uma ferramenta para geração de sistemas interativos", Tese de Mestrado, UNICAMP-FEC-DEE, maio-1985

/THO-85/ - Thomas, J.J.- "Architecture for a User Interface Management System"- Proceedings of the Workshop on User Interface Management Systems; novembro 1-3, 1983; Seeheim; pp 81/85

10.2- BIBLIOGRAFIA ADICIONAL

/COU-85/ - Coutaz, Joelle - "Abstractions for User Interface Design", Computer, setembro- 1985

/END-84/ - Enderle, G. - "The Interface of the UIMS to the Application", Computer Graphics Forum, vol 3, 1984

/END-85/ - Enderle, G. - "Computer Graphics Standards", Comput & Graphics, vol 9, número 1, 1985

/FOL-84A/ - Foley, J. D.; Wallace, V. L.; Chan, Peggy - "The Human Factors of Computer Graphics Interaction Techniques", IEEE Computer Graphics and Applications, novembro-1984

/FOL-84B/ - Foley, J. D. - "Human Factors in Computer Systems: A Management Perspective", The 15th Annual Conference and Exposition, Ahaheim, CA, USA, maio-1984, National Computer Graphics Association

/GRE-84/ - Green, Mark - "Report on Dialogue specification Tools", Computer Graphics Forum, vol 3, 1984

/HER-84/ - Herman, I. - "Managing Multiple Context Frames trough GKS", Computer Graphics Forum, vol. 3, 1984

/MAS-84/ - Mason, R. E. A.; Carey, T. T. - "Prototyping Interactive Information Systems", Communications of the ACM, maio-1983

/MYE-84/ - Myers, Brad A. - "The User Interface for Sapphire"- IEEE Computer Graphics and Applications, dezembro-1984

/OLSEN-83/- Olsen Jr., Dan R.- "Automatic Generation of Interactive Systems", Computer Graphics, janeiro-1983

/ROS-83/ - Rosenthal, D. S. H. - "Managing Graphical Resources", Computer Graphics, janeiro-1983

/SIL-84/ - Silva, M. V.; Magalhães, L. P. - "Processadores Universais de Diálogo: Formas de Representação do Diálogo", XI Seminário Integrado de Software e Hardware, IV Congresso da Sociedade Brasileira de Computação, Viçosa, MG, 21 a 27 de julho de 1984

/SILVA-85/- Silva, Sidney D. - "Uma Ferramenta para Apoio ao Processo de Projeto de Diálogos e Sistemas Interativos", Tese de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, agosto-1985

/STR-84/ - Strubbe, H. J. - "Role, Model, Structure and Construction of a UIMS", Working Group Report, Computer Graphics Forum, vol.3, 1984

/TAK-85/ - Takala, Tapio - "User Interface Management System with Geometric Modeling Capability: A CAD System's Framework", IEEE Computer Graphics, janeiro-1983

APENDICE A: DESCRIÇAO FUNCIONAL DO GAV

Neste apêndice, as funções do GAV (exceto as funções inquire para a sua estrutura de dados e as funções que permitem a utilização dos recursos GKS em cada PV) são descritas de uma maneira semelhante àquela utilizada pela norma GKS, para descrever as suas funções.

Nesta descrição constam:

1- nome da função

2- parâmetros

- tipo: entrada (E)

  saída (S)

- nome

- tipo de dado: inteiro (I)

  real (R)

  nome (N)

  ponto (P)

3- descrição do efeito causado pela função

4- erros : identifcador

  mensagem

OPEN GAV

PARAMETROS

E	NOME DO ARQUIVO GERAL DE ERROS	1

EFEITO: A função do GKS 'OPEN GKS' é chamada. O GAV é colocado no estado de operação 'ABERTO' e sua estrutura de dados é inicializada sendo realizadas as seguintes definições:

1) Uma ET OUTIN para cada tipo previsto na implementação, com área de visualização igual à tela útil;
2) Uma ET pertencente à categoria MO, outra à categoria MI e uma terceira pertencente à categoria WISS;
3) O PV 1 (PV default ao início) ao qual são associadas as ET's definidas nos itens 1 e 2;
4) Uma CT para cada tipo de ET OUTIN existente. A cada uma delas é associda uma ET OUTIN de mesmo tipo definida no item 1.

O PV definido no item 3 é estabelecido como o PV corrente e as CT's definidas no item 4 têm seus identificadores armazenados na entrada 'configurações de tela correntes' da tabela 4.7 da estrutura de dados do GAV. O arquivo geral de erros é aberto e seu nome é armazenado na tabela 4.7 da estrutura de dados do GAV.

ERROS:

2026- O nome do arquivo de erro é inválido
2028- O GAV já se encontra no estado de operação 'ABERTO'

CLOSE GAV

PARAMETROS

nenhum

EFEITO: O GAV é colocado no estado de operação 'FECHADO' e é chamada a função do GKS 'CLOSE GKS'.

ERROS:

2000- O GAV não está aberto
2029- O estado do GKS para algum PV é diferente de GKCL

DEFINE ET

PARAMETROS

E	identificador da ET	N
E	identificador de conexão	I
E	tipo	I
E	área de visualização	2xP
E	prioridade	I
E	índice de cor	I

EFEITO: Define uma ET. Se a categoria da ET especificada é OUTIN, os dados (identificador da ET, identificador de conexão, tipo, área de visualização, prioridade e índice de cor) são armazenados na tabela 4.1 da estrutura de dados do GAV. Caso contrário, os dados (neste caso, apenas o identificador da ET, o identificador de conexão e o tipo) são armazenados na tabela 4.2 da estrutura de dados do GAV.

OBS: A redefinição de uma estação de trabalho aberta, pertencente à CT corrente, ocasiona:

1)o apagamento da tela útil;
2)modificação da viewport da transformação de workstation;
3)regeneração do conteúdo atualizado da ET em questão (através da função 'REDRAW ALL SEGMENTS ON WORKSTATION');
4)regeneração do conteúdo não atualizado das demais ET's abertas que compõem a CT corrente em questão.

ERROS:

20- O identificador de ET especificado é inválido
21- O identificador de conexão especificado é inválido
22- O tipo especificado é inválido
85- Índice de cor é inválido
2000- O GAV não está aberto
2005- Categoria não é aceita pelo GAV
2006- A área de visualização da ET está fora da tela útil
2008- Prioridade é inválida
2009- Número máximo de ET's OUTIN que podem ser definidas já foi atingido
2010- ET's pertencentes às categorias MO, MI, OUTPUT e WISS não podem ser redefinidas
2011- ET pertencente à categoria WISS já foi definida
2012- Número máximo de ET's MO, MI, OUTPUT ou WISS que podem ser definidas já foi atingido

DEFINE PV

PARAMETROS

E	identificador do PV	I
E	número de ET's associadas	I
E	identificadores das ET's associadas	nxN

EFEITO: Define um PV. O identificador do PV e os identificadores das ET's a ele associadas são armazenados na tabela 4.4 da estrutura de dados do GAV.

OBS: Quando um PV é redefinido, as ET's que deixam de estar a ele associadas são desativadas e fechadas. As CT's correntes que contêm em seu conjunto de ET's associadas alguma(s) dessas ET's e onde ainda, ocorre superposição das áreas de visualização, são regeneradas.

ERROS:

2000- GAV não está aberto
2001- ET não foi definida
2013- O identificador do PV é inválido
2014- ET já está associada a outro PV e não pertence à categoria WISS
2015- O número máximo de PV's que podem ser definidos já foi atingido
2025- O estado do GKS para o PV deve ser: GKCL, GKOP, WSOP ou WSAC
2030- N é maior que o número máximo de ET's que podem estar associadas a um PV

DEFINE CT

PARAMETROS

E	identificador da CT	I
E	número de ET's que a compõem	I
E	ET's que a compõem	nxN

EFEITO: Define uma CT. O identificador da CT, os identificadores das ET's que a compõem e o tipo ao qual elas pertencem são armazenados na tabela 4.3 da estrutura de dados do GAV.

OBS: A redefinição de uma CT corrente ocasiona:

1) apagamento da tela útil;
2) regeneração do conteúdo atualizado das ET's que passaram a fazer parte da CT em questão (através da função do GKS 'REDRAW ALL SEGMENTS ON WORKSTATION');
3) regeneração do conteúdo não atualizado das ET's que permanecem compondo a CT em questão (através da leitura dos arquivos de tela).

ERROS:

2000- O GAV não está aberto
2001- ET não foi definida
2016- O identificador da CT é inválido
2017- N é maior que o número máximo de ET's que podem compor uma CT
2018- ET não pertence à categoria OUTIN
2023- ET's não são do mesmo tipo
2024- Número máximo de CT's que podem ser definidas já foi atingido

SET PV CORRENTE

PARAMETROS

E	identificador do PV	I

EFEITO: Define um PV como o PV corrente. Desativa ET's cujos identificadores encontrem-se armazenados no conjunto de ET's ativas do antigo PV corrente. Atualiza a entrada 'processo visual corrente' da tabela 4.7 da estrutura de dados do GAV. Copia dados relativos ao novo PV corrente da tabela 4.4 da estrutura de dados do GAV na estrutura de dados do GKS (utilizando as funções 'SET' do GKS). Ativa ET's cujos identificadores encontem-se armazenados no conjunto de ET's ativas do novo PV corrente.

ERROS:

2000- O GAV não está aberto
2003- PV não foi definido
2013- O identificador do PV é inválido
2025- O estado do GKS para o PV deve ser: GKCL, GKOP, WSOP ou WSAC

SET CT CORRENTE

PARAMETROS

E	identificador da CT	I

EFEITO: Define uma CT como a CT corrente. Atualiza a entrada 'configurações de tela correntes' para o tipo em questão, da tabela 4.7 da estrutura de dados do GAV. Limpa a tela útil. Regenera o conteúdo atualizado das ET's que a compõem (utilizando a função do GKS 'REDRAW ALL SEGMENTS ON WORKSTATION').

ERROS:

2000- O GAV não está aberto
2002- CT não foi definida
2016- O identificador da CT é inválido
2002- CT não foi definida

SET PRIORIDADE

PARAMETROS

E	identificador da ET	I
E	prioridade	I

EFEITO: Modifica a prioridade de uma ET, estabelecida no momento de sua definição.

OBS: A alteração da prioridade associada a uma ET pertencente a uma CT corrente ocasiona:

1) apagamento da tela útil;
2) regeneração do conteúdo não atualizado das ET's que compõem a CT corrente (através da leitura dos arquivos de tela das ET's).

ERROS:

2000- O GAV não está aberto
2004- A ET não está associada ao PV corrente
2008- Prioridade é inválida
2018- A ET não é OUTIN

APENDICE B: IMPLEMENTAÇÃO DO EXEMPLO DE UTILIZAÇÃO DO GAV

Este apêndice apresenta o resultado da implementação do exemplo de utilização do GAV, descrito no capítulo 8. Uma visão geral do interrelacionamento dos módulos que compõem o programa, é fornecida na figura B.1.

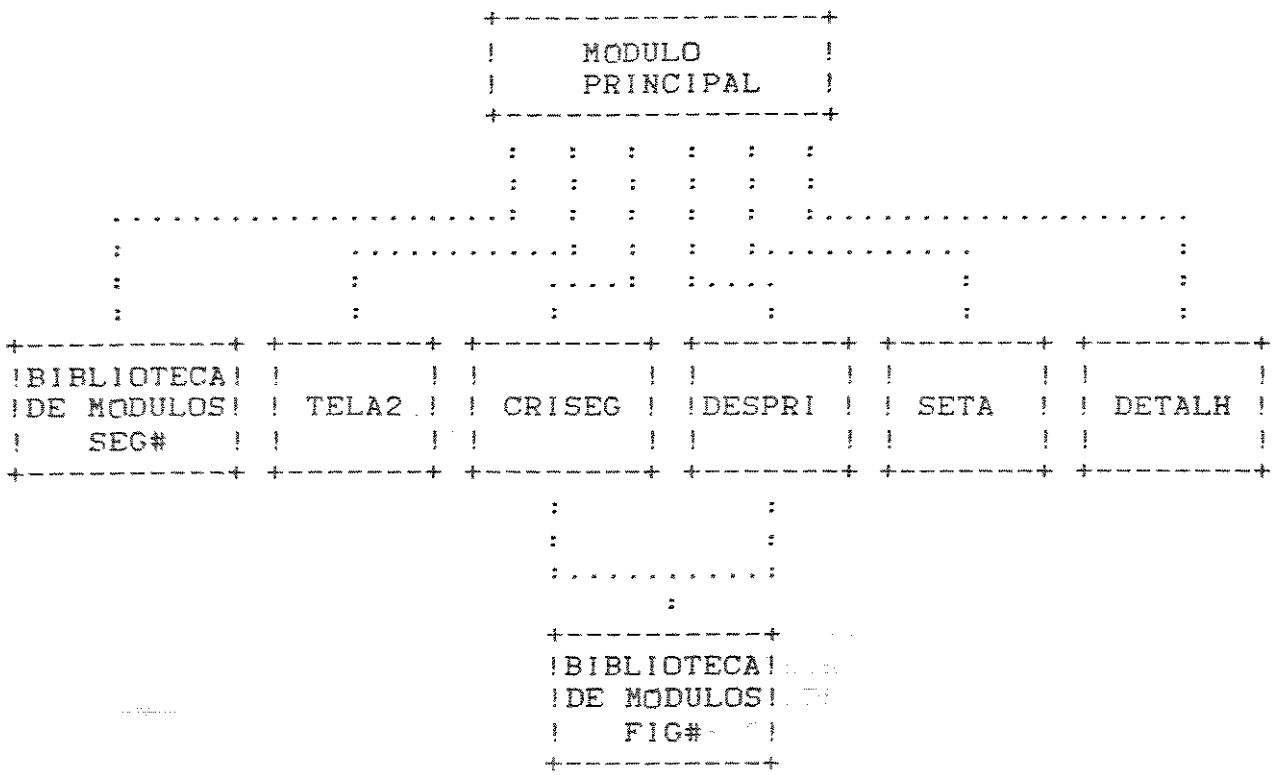

fig.B.1- Estrutura modular do programa exemplo de utilização do GAV

```fortran
c***********************************************************************
c*                                                                     *
c*                                                                     *
c*              EXEMPLO DE APLICACAO DO GAS                            *
c*                                                                     *
c*                                                                     *
c***********************************************************************
c
      implicit none
c
      include 'contos.cmm'
c
c ----------- declaracao das variaveis locais ---------------------
c
      integer           etas1(7),etas2(5),i,asf(13),sta,ON,OFF,
     *                  ist1,ist2,nicc,sg,tn,etas3(3),chn,OK,
     *                  NONE,NOPICK,inder,nt,pri,sta6,len,CONT,
     *                  FIRST,SEC,THIRD

      real              ppx(2),ppy(2),ponx,pony,posx,posy
c
      character*13      tex
c
      character*1       str15
c
      character*10      tex1
c
c ----------- inicializacao ---------------------------------------
c
      do i=1,13
        asf(i)=1          ! INDIVIDUAL
      end do
      data  ponx/.5/
      data  pony/.3/
c
      FIRST=0
      SEC=0
      THIRD=0
      OFF=0
      ON=1
      OK=1
      NONE=0
      NOPICK=0
      CONT=10           ! contador utilizado para nome de seg
c
c ----------- abre o GAS ------------------------------------------
c
      call opgav(65)
c
c ----------- define estacoes de trabalho -------------------------
c
      call dfnet(15,5,24,2,0.,.19625,.03875,.155,1,2)
      call dfnet(16,6,24,2,.19625,.335,.03875,.155,1,2)
      call dfnet(17,7,24,2,.0,.19625,.0,.03875,1,2)
      call dfnet(18,8,24,2,.19625,.335,.0,.03875,1,2)
      call dfnet(19,9,24,2,.145,.195,.05,.0766,1,2)
c
c ----------- define  processo visual -----------------------------
c                                                                  B1
      do i=1,5
```

```
          etas1(i)=i+1      ! ET's 2 a 6
        end do
        etas1(6)=11
        etas1(7)=14
c
        etas3(1)=12
        etas3(2)=13
        etas3(3)=14
        call dfnav(1,3,etas3) ! desassocia a ET 11 do PN1
        call dfnav(1,7,etas1)
c
c ------------ define configuracoes da tela ------------------------
c
        do i=1,5
         etas2(i)=i+1
        end do
        call dfncr(1,5,etas2)
c
c ------- seta PN corrente ------------------------------------------
c
        call setpvc(2)
c
c ------- acre GKS  e trata menu de entrada -------------------------
c
        call xgopks(15)
        call xgsesf(esf)
        call xgopwk(11,11,24)
        call xgetwk(11)
        call segl
        call segt
        call xgcewk(11)
c
c -------- obtem opcao desejada -------------------------------------
c
  100   continue
        sta=NONE
        do while (sta.NE.OK)
         call xgreqch(11,41,sta,chn)
        end do
c
c ---- apresenta opcao
c
        if(chn.GE.1.AND.chn.LE.4)
     #    then
            str1[ecner(chn+48)
c
            call xgacwk(11)
            call xcstxc(2)
            call xctx(ponx,pony,str15)
            call xgstxc(1)
            call xgcwuk(11)
        endif
c
        if(chn.EQ.2) then           ! MANIPULACAO
           go to 200
        else if(chn.EQ.3) then      ! HELP
           go to 300
        else if(chn.EQ.4) then      ! SAIDA
           go to 400
        else if(chn.NE.1.AND.chn.NE.4) then   ! pedir nova opcao
```

```
            go to 100
         endif
c
c -------- OPCAO: DEFINICAO ------------------------------------
c
         call tela2(tex1)
c
c --------- seta nova CT corrente -------------------------------
c
         call setctc(2)
c
c -------- abre demais ET's ------------------------------------
c
         if (FIRST.EQ.0)
     *     then
            call xgopwk(14,14,90)
            call xgopwk(15,5,24)
            call xgopwk(16,6,24)
            call xgopwk(17,7,24)
            call xgopwk(18,8,24)
            call xgswkw(17,0.,1.0,0.,0.2)
c
c -------- traca nome da base de dados na ET 17 ----------------
c
         endif
         call xgswn(1,0.,50.,0.,10.)
         call xgsvp(1,0.,1.0,0.,0.2)
         call xgseln(1)
         call xgacwk(17)
         call seg3(tex1)
         if(FIRST.EQ.0)
     *     then
            call seg6
            call seg7
            call seg8
            call seg9
            call seg15
            call seg16
         endif
         call xgcawk(17)
         call xgseln(0)
c
c -------- traca menu de acoes centro da definicao -------------
c
         if(FIRST.EQ.0)
     *     then
            call xgacwk(16)
            call seg4
            call seg5
            call xgcawk(16)
c
c -------- traca menu principal --------------------------------
c
            call xgswn(1,0.,10.0,0.,30.0)
            call xgsvp(1,0.,.25,0.,.75)
            call xgseln(1)
            call xgswkw(16,0.,.25,0.,.75)
            call xgsln(1)
            call xgsplc(3)
            call xgacwk(16)
```

```
c
c              call crsreg
c
               call xgrswk(15)
            endif
c
c ---------- obter acao desejada ------------------------------------------
c
 200        continue
            sta=NONE
            do while (sta.NE.OK)
             call xgrqcn(15,41,sta,chn)
            end do
            if(chn.EQ.2) then              ! SAIDA
               call xgsvis(1,on)    ! DEVE RETORNAR AO MENU PRINC
               call xgseln(0)
               call setcts(1)
               call xgclru(15,1)
               call xgdsgu(17,3)    ! APAGA SEG. NOME DA BASE DE DADOS
               FIRST=1
               if(THIRD.EQ.1)
     *           then
                  call xgclwk(19)
                  THIRD=0
               endif
               FIRSEG=0
               go to 100
            endif
c
c ---------- OPCAO:CRIAR --------------------------------------------------
c
c ---------- obtem primitiva desejada -------------------------------------
c
            call xgsvis(6,OFF)    ! ESCREVE PROMPT " ESCOLHA TIPO DE
            call xgsvis(16,ON)    ! PRIMITIVA" NA ET 17
            call xgrswk(17)
            sta=NOPICK
            do while (sta.NE.OK)
              call xgrqpk(16,51,sta,sg,picc)
            end do
c
            if(picc.EQ.1) then
c
c ------- apresentar menu 1 -------------------------------------------------
c
                  call xgsvis(10,OFF)
                  call xgsvis(11,ON)
c
            else if(picc.EQ.2) then
c
c --------- apresentar menu2 -------------------------------------------------
c
                  call xgsvis(10,OFF)
                  call xgsvis(12,ON)
c
            else if(picc.EQ.3) then
c
c ---------- apresentar menu 3 -------------------------------------------------
c
                  call xgsvis(10,OFF)
```

```
                call xgsvis(18,OFF)
c
        else if(pick.EQ.4) then
c
c  ---------- apresenta menu 4 ----------
c
                call xgsvis(10,OFF)
                call xgsvis(14,ON)
c
        endif
        call xgrswk(16)
c
c  ---------- apresenta novo menu da ET 18 ----------
c
        call xgsvis(4,OFF)
        call xgsvis(5,ON)
        call xgrswk(18)
c
c  ---------- abre ET 19 ----------
c
        if (THIRD.EQ.0)
     *     then
             call xgopwk(19,9,24)
             THIRD=1
        endif
c
c  ---------- obtem opcao desejada ----------
c
        call xgsvis(16,OFF)             ! ESCREVE PROMPT "ESCOLHA
        call xgsvis(6,ON)               ! OPCAO" NA ET 17
        call xgrswk(17)
        SEC=0
  300   continue
        sta=NONE
        do while (sta.NE.OK)
          call xgrqch(18,41,sta,chn)
        end do
        if(chn.EQ.2) then               ! REMOVER
           go to 400
        else if(chn.EQ.3) then          ! ALTERAR
           go to 400
        else if(chn.EQ.4) then          ! DETALHAR
           call detaln
           go to 300
        else if(chn.EQ.5) then          ! LISTAR
           go to 400
        else if(chn.EQ.6) then          ! SAIR
           call xgsvis(10,ON)       ! ET 16
           call xgsvis(11,OFF)
           call xgsvis(12,OFF)
           call xgsvis(13,OFF)
           call xgsvis(14,OFF)
           call xgrswk(16)
           call xgsvis(4,ON)        ! ET 18
           call xgsvis(5,OFF)
           call xgrswk(18)
           call xgseln(0)
           go to 200
        else if(chn.LT.1.OR.chn.GT.6) then ! NOVO CHOICE
           go to 300
```

```
      endif
c
c ------------ escolha p/ def ------------------------------------------
c
      if(SEC.EQ.0)
     *   then
           call xgsun(1,0.,.2,0.,.8,0)
           call xgsvp(1,0.,.3,0.,.2)
           call xgswkw(15,0.,.3,0.,.2)
           call xgsvpi(0,1,1)                    ! SETA PRIORIDADE
           SEC=1
      endif
      call xgseln(1)
      call xgacwk(15)
c
c ---------- escreve prompt ------------------------------------------
c
      call xgsvis(6,OFF)    ! ESCREVE PROMPT " ESCOLHA
      call xgsvis(7,ON)     ! PRIMITIVA" NA ET 17
      call xgrswk(17)
c
c ------------ obtem primitiva -----------------------------------------
c
      sta=NOPICK
      do while (sta.NE.OK)
        call xgrqpk(16,51,sta,sg,picc)          ! OBTEM PRIM
      end do
c
c ------------ escreve prompt ------------------------------------------
c
      call xgsvis(7,OFF)    ! ESCREVE PROMPT "POSICIONE
      call xgsvis(8,ON)     ! CURSOR" NA ET 17
      call xgrswk(17)
c
c ------------ obtem posicao -------------------------------------------
c
      sta=NONE
      do while (sta.NE.OK)
        call xgrqlc(15,11,sta,tn,ppx(1),ppy(1))  ! OBTEM POSICAO
      end do
c
      posegx(CONT-16)=ppx(1)                     ! GUARDA POSICAO
      posegy(CONT-16)=ppy(1)
c
      if(picc.EQ.4.OR.picc.EQ.5) then            ! reta ou seta
         segtip(CONT-16)=4
      else if(sg.EQ.11) then                     ! entidades
         segtip(CONT-16)=1
      else if(sg.EQ.12) ther                     ! relacionamentos
         segtip(CONT-16)=2
      else if((sg.EQ.13.OR.sg.EQ.14).AND.picc.EQ.1) then !esquemas
         segtip(CONT-16)=1                                !agrup.
      else if((sg.EQ.13.OR.sg.EQ.14).AND.picc.EQ.2) then !esquemas
         segtip(CONT-16)=3                                !agrup.
      endif
c
      if(picc.EQ.4.OR.picc.EQ.5)
     *   then
           sta=NONE
           do while (sta.NE.OK)                                    B7
```

```
            call xgrqlc(15,11,sta,tn,posx(1),posy(1)) ! LOCATOR 1 ET.
          end do
        endif

c
c ---------- desenha primitiva na ET 15 ----------------------------
c
        call xgacwk(15)                 ! ATIVA ET WISS
        call xgqrsg(CONT)
        if(picc.EQ.5)
     *    then
            call seta(ppx,ppy)
          else
            call desori(sg,picc,ppx,ppy)
        endif

c
c ---------- escreve prompt ----------------------------------------
c
        if(picc.NE.--.AND.picc.NE.5)
     *    then
            call xgsvis(3,OFF)      ! ESCREVE PROMPT "ENTRE TEXTO"
            call xgsvis(3,ON)       ! NA ET 17
            call xgrswk(17)

c
c -------- obtem texto da primitiva ------------------------------
c
            sta=NONE
            do while (sta.NE.OK)
              call xgrost(15,el,sta,len,tex)
            end do

c
c ------- escreve prompt ----------------------------------------
c
            call xgsvis(3,OFF)      ! ESCREVE PROMPT "POSICIONE
            call xgsvis(3,ON)       ! CURSOR" NA ET 17
            call xgrswk(17)

c
c ---------- obtem posicao do texto --------------------------------

            sta=NONE
            do while (sta.NE.OK)
             call xgrqlc(15,11,sta,tn,posx,posy)
            end do

c
c ------- gera texto na posicao obtida --------------------------
c
            call xgtx(posx,posy,tex)

c
        endif

c
c ---------- fecha segmento e desativa ET 15 ---------------------
c
        call xgclsg
        call xgdawk(15)
        call xgdawk(14)                 ! DESATIVA ET WISS

c
c ---------- incrementa contador de segmentos --------------------
c
        CONT=CONT+1

c
c --------- escreve prompt e obtem nova opcao ---------------------
```

```
c
      call xgsvis(9,OFF)
      call xgsvis(6,ON)
      call xerrsw(17)
      go to 300
c
c ---------- Fix ----------
c
  400 continue
      end
```

```
c ==========  subrotina detalh  ==========================================
c
c finalidade : apresenta na ET 16 um detalhe do estudo observado na ET
c              15, escolhido pelo usuario
c
c linguagem: FORTRAN-77
c
c ==========================================================================
c
      subroutine detalh
      implicit none
c
c ---------- inclui arquivos -----------------------------------------------
c
      include 'pontos.cmm'
c
c -------- declaracao de variaveis -----------------------------------------
c
      integer       status,segnam,pick,OK,OFF,ON,NONE
c
      real          mt(2,3),trx,try,xx,yy
c
c -------- inicializacao ---------------------------------------------------
c
      OK=1
      OFF=0
      ON=1
      NONE=0
c
c -------- apresenta prompt na ET 17 ---------------------------------------
c
      call xgsvis(6,OFF)
      call xgsvis(15,ON)     ! ESCOLHA DETALHE
      call xgrswk(17)
c
c -------- seta TN corrente e window da TW ---------------------------------
c
      call xgseln(1)
      if(FIRSEG.EQ.0)
     *   then
           call xgswkw(19,.0,.3,.0,.2)
      endif
c
c ---------- obtem detalhe ---------------------------------------------------
c
  100 continue
      status=NONE
      do while (status.NE.OK)
       call xgrqpk(15,51,status,segnam,pick)
      end do
c
c ---------- verifica se e reta ou seta --------------------------------------
c
      if(segtab(SEGNAM-16).EQ.4)
     *   then
           go to 100  ! ESCOLHER NOVO DETALHE
      endif
c
c ---------- calcula matriz de transformacao de seg. -------------------- BlO
c
```

```
      if(segtip(segnam-16).EQ.1)  then
          xx=posegx(segnam-16)
          yy=posegy(segnam-16)
          trx=-(xx-9.0)
          try=-(yy-5.0)
      else if(segtip(segnam-16).EQ.2) then
          xx=posegx(segnam-16)-3.0
          yy=posegy(segnam-16)
          trx=-(xx-9.0)
          try=-(yy-2.0)
      else if(segtip(segnam-16).EQ.3) then
          xx=posegx(segnam-16)
          yy=posegy(segnam-16)
          trx=-(xx-5.0)
          try=-(yy-2.0)
      endif
c
      call gevtm(xx,yy,trx,try,0.,4.,4.,0,mt)

c ------- seta matriz de transformacao de seg. ------------------------
c
      call xgssgt(segnam,mt)
c
c ------- limpa ET 19  se ja existe detalhe ---------------------------
c
      if(FIRSEG.EQ.1)
     *   THEN
           call xgclrw(19,1)
      endif
c
c ------- associa seg. a ET 19 ----------------------------------------
c
      call xgasgw(19,segnam)
c
      FIRSEG=1
c
c ------- apresenta prompt na ET 17 -----------------------------------
      call xgsvis(15,OFF)
      call xgsvis(6,ON)        ! ESCOLHA OPCAO
      call xgrswk(17)
c
c -------- FIM ---------------------------------------------------------
c
      return
      end
```

```
c ========== subrotina tela2 ==============================
c
c finalidade: obter nome da base de dados,versao,autor e data
c
c linguagem: FORTRAN-77
c
c=============================================================
c
      subroutine tela2(txt1)
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer      len1,len2,len3,len4,sta,off,on,NONE,OK
c
      real         ppx(4),ppy(4)
c
      character*10 txt1
c
      character*4  txt2
c
      character*20 txt3
c
      character*8  txt4
c
      character*1  txx
c
c --------- inicializacao -----------------------------------
c
      OFF=0
      ON=1
      NONE=0
      OK=1
      data ppx/.55,.45,.45,.6/
      data ppy/.7,.6,.5,.4/
c
c -----------------------------------------------------------
c
      call xgacwk(11)
      call xgstxc(2)
      call xgsvis(1,OFF)
      call xgsvis(2,ON)
      call xgrswk(11)
      call xgsvis(2,OFF)
c
c ------ obtem nome da base de dados --------------------------
c
  100 continue
      sta=NONE
      do while (sta.NE.OK)
        call xgrqst(11,61,sta,len1,txt1)
      end do
      if(len1.LT.1.OR.len1.GT.10)
     *     then
            go to 100
          else
            call xgtx(ppx(1),ppy(1),txt1)
      endif
c
c ---------- obtem versao ------------------------------------
```

```
c
  200   continue
        sta=NONE
        do while (sta.NE.OK)
           call xgrqst(11,61,sta,len2,txt2)
        end do
        if(len2.LT.1.OR.len2.GT.4)
     *    then
            go to 200     ! NOVO PEDIDO
          else
            call xgtx(ppx(2),ppy(2),txt2)
        endif
c
c -------- obtem autor ------------------------------------------
c
  300   continue
        sta=NONE
        do while (sta.NE.OK)
           call xgrqst(11,61,sta,len3,txt3)
        end do
        if(len3.LT.1.OR.len3.GT.20)
     *    then
            go to 300     ! NOVO PEDIDO
          else
            call xgtx(ppx(3),ppy(3),txt3)
        endif
c
c ---------- obtem data ------------------------------------------
c
  400   continue
        sta=NONE
        do while (sta.NE.OK)
          call xgrqst(11,61,sta,len4,txt4)
        end do
        if(len4.NE.8)
     *    then
            go to 400
          else
            call xgtx(ppx(4),ppy(4),txt4)
        endif
        accept 450,txx
  450   format(a1)
        call xgstxc(1)
        call xgdewk(11)
c
c ----------------------------------------------------------------
c
        return
        end
```

```
c ========== subrotina despri ==========================================
c
c finalidade : desenha primitiva na ET 15
c
c linguagem: FORTRAN-77
c
c ========================================================================
c
      subroutine despri(segno,pick,posx,posy)
      implicit none
c
c ---------- declaracao de variaveis ------------------------------------
c
      integer     segno,pick
c
      real        posx(2),posy(2)
c
c ------------------------------------------------------------------------

      if(segno.EQ.11) then
         if(pick.EQ.1) then
            call fig1(posx(1),posy(1))
         else if(pick.EQ.2) then
            call fig5(posx(1),posy(1))
         else if(pick.EQ.3) then
            call fig6(posx(1),posy(1))
         else if(pick.EQ.4) then
            call xgpl(2,posx,posy)
         else if(pick.EQ.5) then
            call xgpl(2,posx,posy)
         endif
      else if(segno.EQ.12) then
         if(pick.EQ.1) then
            call fig2(posx(1),posy(1))
         else if(pick.EQ.2) then
            call fig9(posx(1),posy(1))
         else if(pick.EQ.3) then
            call fig10(posx(1),posy(1))
         else if(pick.EQ.4) then
            call xgpl(2,posx,posy)
         else if(pick.EQ.5) then
            call xgpl(2,posx,posy)
         endif
      else if(segno.EQ.13) then
         if(pick.EQ.1) then
            call fig3(posx(1),posy(1))
         else if(pick.EQ.2) then
            call fig11(posx(1),posy(1))
         else if(pick.EQ.4) then
            call xgpl(2,posx,posy)
         else if(pick.EQ.5) then
            call xgpl(2,posx,posy)
         endif
      else if(segno.EQ.14) then
         if(pick.EQ.1) then
            call fig4(posx(1),posy(1))
         else if(pick.EQ.2) then
            call fig12(posx(1),posy(1))
         else if(pick.EQ.4) then
            call xgpl(2,posx,posy)
```

```
            else if(pick.EQ.5) then
               call xgpl(2,posx,posy)
            endif
         endif
C
         return
         end
```

UNICAMP

```
c ========== subrotina criseg ==========================================
c
c finalidade : abre cada uma das ET, ativa-a e cria segmentos invisiveis
c
c linguagem: FORTRAN-77
c
c ========================================================================
c
      subroutine criseg
      implicit none
c
c -------- declaracao de variaveis -------------------------------
c
      integer      off
c
c -------- inicializacao -----------------------------------------
c
      off=0

c ------------------------------------------------------------------
c
      call xgcrsg(10)                ! menu princ prim
      call xgspki(1)
      call fig1(2.0,23.5)
      call xgspki(2)
      call fig2(5.0,15.5)
      call xgspki(3)
      call fig3(2.0,9.5)
      call xgspki(4)
      call fig4(2.0,3.5)
      call xgclsg
c
      call xgcrsg(11)
      call xgsvis(11,off)
      call xgspki(1)
      call fig1(2.0,23.5)
      call xgspki(2)
      call fig5(2.0,17.5)
      call xgspki(3)
      call fig6(2.0,11.5)
      call xgspki(4)
      call fig7(2.0,8.0)
      call xgspki(5)
      call fig8(5.0,7.0)
      call xgclsg
c
      call xgcrsg(12)
      call xgsvis(12,off)
      call xgspki(1)
      call fig2(5.0,23.0)
      call xgspki(2)
      call fig9(5.0,16.0)
      call xgspki(3)
      call fig10(5.0,9.0)
      call xgspki(4)
      call fig7(2.0,5.5)
      call xgspki(5)
      call fig8(5.0,4.0)
      call xgclsg
c
```

```
      call xgcrsg(13)
      call xgsvis(13,off)
      call xgspki(1)
      call fig3(2.0,23.0)
      call xgspki(2)
      call fig11(1.0,14.0)
      call xgspki(4)
      call fig7(2.0,9.0)
      call xgspki(5)
      call fig8(5.0,7.0)
      call xgclsg
c
      call xgcrsg(14)
      call xgsvis(14,off)
      call xgspki(1)
      call fig4(2.0,23.0)
      call xgspki(2)
      call fig12(1.0,14.0)
      call xgspki(4)
      call fig7(2.0,9.0)
      call xgspki(5)
      call fig8(5.0,7.0)
      call xgclsg
c
c ------------------------------------------------------------------------
c
      return
      end
```

UNICAMP

```
c ========== subrotina seta ==============================
c
c finalidade: calcular pontos e desenhar seta
c
c linguagem: FORTRAN-77
c
c ========================================================
c
      subroutine seta(posx,posy)
      implicit none
c
c -------- declaracao de variaveis ---------------------
c
      real posx(2),posy(2),teta,dx,dy,dx1,dy1,px(3),
     *     py(3),teta1
c
c ------------------------------------------------------
c
      teta=ATAN2D((posy(2)-posy(1)),(posx(2)-posx(1)))
      teta1=teta
      if(teta.LT.0)
     *  then
          teta=360+teta
      endif
      dx=COSD(teta-45)
      dy=SIND(teta-45)
      dx1=COSD(135-teta)
      dy1=SIND(135-teta)
c
      px(1)=posx(1)-dx1
      py(1)=posy(1)+dy1
c
      px(2)=posx(1)
      py(2)=posy(1)
c
      px(3)=posx(1)+dx
      py(3)=posy(1)+dy

      call xgpl(2,posx,posy)
      call xgpl(3,px,py)
c
      return
      end
```

```
c ========== subrotina seg1 ==============================
c
c finalidade:criar o segmento 1
c
c linguagem: FORTRAN-77
c
c ========================================================
c
      subroutine seg1
      implicit none
c
c --------- declaracao de variaveis -----------------------
c
      real              ponx(5),pony(5)
c
      character*11      str1
c
      character*13      str2

      character*6       str3
c
      character*7       str4
c
      character*27      str5
c
c --------- inicializacao ---------------------------------
c
      data  ponx/.4,.4,.4,.4,.3/
      data  pony/.7,.6,.5,.4,.3/
      str1='1-DEFINICAO'
      str2='2-MANIPULACAO'
      str3='3-HELP'
      str4='4-SAIDA'
      str5='ENTRE COM A OPCAO DESEJADA:'
c
c ---------------------------------------------------------
c
      call xgcrsg(1)
      call xgtx(ponx(1),pony(1),str1)     ! DEFINICAO
      call xgtx(ponx(2),pony(2),str2)     ! MANIPULACAO
      call xgtx(ponx(3),pony(3),str3)     ! HELP
      call xgtx(ponx(4),pony(4),str4)     ! SAIDA
      call xgtx(ponx(5),pony(5),str5)! ENTRE COM A OPCAO DESEJADA
      call xgclsg
c
      return
      end
```

```
c ============ subrotina seg2 ================================
c
c finalidade:criar o segmento 2
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg2
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          off
c
      real             ponx(4),pony(4)
c
      character*13     str1

      character*7      str2
c
      character*6      str3
c
      character*16     str4
c
c --------- inicializacao -----------------------------------
c
      data  ponx/.3,.3,.3,.3/
      data  pony/.7,.6,.5,.4/
      str1='NOME ESQUEMA:'
      str2='VERSAO:'
      str3='AUTOR:'
      str4='DATA (DD/MM/AA):'
      off=0
c
c ------------------------------------------------------------
c
      call xgcrsg(2)
      call xgsvis(2,off)
      call xgtx(ponx(1),pony(1),str1)    ! NOME ESQUEMA
      call xgtx(ponx(2),pony(2),str2)    ! VERSAO
      call xgtx(ponx(3),pony(3),str3)    ! AUTOR
      call xgtx(ponx(4),pony(4),str4)    ! DATA
      call xgclsg
c
      return
      end
```

UNICAMP

```
c ============ subrotina seg3 ================================
c
c finalidade:criar o segmento 3
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg3(strr)
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      real              ponx(2),pony(2)
c
      character*10      strr
c
      character*22      str1

c --------- inicializacao -----------------------------------
c
      data  ponx/5.,22./
      data  pony/8.,8./
      str1='NOME DA BASE DE DADOS:'
c
c -----------------------------------------------------------
c
      call xgcrsg(3)
      call xgtx(ponx(1),pony(1),str1) ! NOME DA BASE
                                      ! DE DADOS:
      call xgtx(ponx(2),pony(2),strr)
      call xgclsg
c
      return
      end
```

```
c ========== subrotina seg4 ================================
c
c finalidade:criar o segmento 4
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg4
      implicit none
c
c --------- declaracao de variaveis -----------------------
c
      real             ponx(2),pony(2)
c
      character*7      str1
c
      character*6      str2

c --------- inicializacao ---------------------------------
c
      data  ponx/.2,.2/
      data  pony/.7,.5/
      str1='1-CRIAR'
      str2='2-SAIR'
c
c ------------------------------------------------------------
c
      call xgcrsg(4)
      call xgtx(ponx(1),pony(1),str1)         ! 1-CRIAR
      call xgtx(ponx(2),pony(2),str2)         ! 2-SAIR
      call xgclsg
c
      return
      end
```

UNICAMP

```
c ========== subrotina seg5 ================================
c
c finalidade:criar o segmento 5
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg5
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          off
c
      real             ponx(6),pony(6)
c
      character*7      str1
      character*9      str2,str3
c
      character*10     str4
c
      character*8      str5
c
      character*6      str6
c
c --------- inicializacao -----------------------------------
c
      data  ponx/.2,.2,.2,.2,.2,.2/
      data  pony/.95,.80,.65,.50,.35,.20/
      str1='1-criar'
      str2='2-remover'
      str3='3-alterar'
      str4='4-detalhar'
      str5='5-listar'
      str6='6-sair'
      off=0
c
c ------------------------------------------------------------
c
      call xgcrsg(5)
      call xgsvis(5,off)
      call xgtx(ponx(1),pony(1),str1)          ! 1-criar
      call xgtx(ponx(2),pony(2),str2)          ! 2-remover
      call xgtx(ponx(3),pony(3),str3)          ! 3-alterar
      call xgtx(ponx(4),pony(4),str4)          ! 4-inserir
      call xgtx(ponx(5),pony(5),str5)          ! 5-listar
      call xgtx(ponx(6),pony(6),str6)          ! 6-sair
      call xgclsg
c
      return
      end
```

UNICAMP

```
c ============ subrotina seg6 ================================
c
c finalidade:criar o segmento 6
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg6
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      real              ponx,pony
c
      character*13      str
c
c --------- inicializacao -----------------------------------

      data  ponx/5./
      data  pony/4./
      str='ESCOLHA OPCAO'
c
c ------------------------------------------------------------
c
      call xgcrsg(6)
      call xgtx(ponx,pony,str) ! ESCOLHA OPCAO
      call xgclsg
c
      return
      end
```

```
c ============ subrotina seg7 ================================
c
c finalidade:criar o segmento 7
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg7
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          off
c
      real             ponx,pony
c
      character*17     str
c
c --------- inicializacao -----------------------------------
c
      data  ponx/5./
      data  pony/4./
      str='ESCOLHA PRIMITIVA'
      off=0
c
c ------------------------------------------------------------
c
      call xgcrsg(7)
      call xgsvis(7,off)
      call xgtx(ponx,pony,str) ! ESCOLHA PRIMITIVA
      call xgclsg
c
      return
      end
```

```
c ========== subrotina seg8 ================================
c
c finalidade:criar o segmento 8
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg8
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          off
c
      real             ponx,pony
c
      character*16     str
c
c --------- inicializacao -----------------------------------
c
      data  ponx/5./
      data  pony/4./
      str='POSICIONE CURSOR'
      off=0
c
c ------------------------------------------------------------
c
      call xgcrsg(8)
      call xgsvis(8,off)
      call xgtx(ponx,pony,str) ! POSICIONE CURSOR
      call xgclsg
c
      return
      end
```

UNICAMP

```
c ========== subrotina seg9 ================================
c
c finalidade:criar o segmento 9
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg9
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          off
c
      real             ponx,pony
c
      character*11     str

c --------- inicializacao -----------------------------------
c
      data  ponx/5./
      data  pony/4./
      str='ENTRE TEXTO'
      off=0
c
c ------------------------------------------------------------
c
      call xgcrsg(9)
      call xgsvis(9,off)
      call xgtx(ponx,pony,str) ! POSICIONE CURSOR
      call xgclsg
c
      return
      end
```

UNICAMP

```
c ============ subrotina seg15 ================================
c
c finalidade:criar o segmento 15
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg15
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          OFF
c
      real             ponx,pony
c
      character*15     str

c --------- inicializacao -----------------------------------
c
      data  ponx/5./
      data  pony/4./
      str='ESCOLHA DETALHE'
      OFF=0
c
c ------------------------------------------------------------
c
      call xgcrsg(15)
      call xgsvis(15,OFF)
      call xgtx(ponx,pony,str) ! ESCOLHA DETALHE
      call xgclsg
c
      return
      end
```

UNICAMP

```
c ============ subrotina seg16 ================================
c
c finalidade:criar o segmento 16
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine seg16
      implicit none
c
c --------- declaracao de variaveis -------------------------
c
      integer          off
c
      real             ponx,pony
c
      character*25     str
c
c --------- inicializacao -----------------------------------
c
      data  ponx/5./
      data  pony/4./
      str='ESCOLHA TIPO DE PRIMITIVA'
      off=0
c
c ------------------------------------------------------------
c
      call xgcrsg(16)
      call xgsvis(16,off)
      call xgtx(ponx,pony,str) ! ESCOLHA TIPO DE PRIMITIVA
      call xgclsg
c
      return
      end
```

```
c  ==========  subrotina  ========  fim2  ==========================
c
c
c  finalidade: fecha o arquivo contendo matriz a ser
c                 lida pelo programa
c
c  linguagem: FORTRAN-77
c
c  ============================================================
c
      subroutine fim2(nx,ny)
      implicit none
c
c  ---------  declaracao de variaveis  -----------------------
c
      real            nx,ny,nx1(5),ny1(5)
c
c  ------------------------------------------------------------
c
      nx1(1)=nx
      ny1(1)=ny
      nx1(2)=nx+0.1
      ny1(2)=ny
      nx1(3)=nx+0.1
      ny1(3)=ny+0.1
      nx1(4)=nx
      ny1(4)=ny+0.1
      nx1(5)=nx
      ny1(5)=ny
c
      call plot(nx1,ny1,5)
c
      return
      end
```

```
c ============ subrotina  fig2 ================================
c
c finalidade: tracar primitiva relacionamento a partir do
c             ponto fornecido
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine fig2(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------------
c
      real          px,py,px1(5),py1(5)
c
c -------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+3.0
      py1(2)=py+3.0
      px1(3)=px
      py1(3)=py+5.0
      px1(4)=px-3.0
      py1(4)=py+1.0
      px1(5)=px
      py1(5)=py
c
      call xyzl(5,px1,py1)
c
      return
      end
```

UNICAMP

```
c ============== subrotina  fig3 ==============================
c
c finalidade: tracar primitiva esquema a partir do ponto
c             fornecido
c
c linguagem: FORTRAN-77
c
c ==========================================================
c
      subroutine fig3(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------------
c
      real          px,py,px1(5),py1(5),px2(3),py2(3),px3(3)
c
c ---------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+6.0
      py1(2)=py
      px1(3)=px+6.0
      py1(3)=py+4.0
      px1(4)=px
      py1(4)=py+4.0
      px1(5)=px
      py1(5)=py
c
      px2(1)=px
      py2(1)=py
      px2(2)=px+1.0
      py2(2)=py+2.0
      px2(3)=px
      py2(3)=py+4.0
c
      px3(1)=px+6.0
      px3(2)=px+5.0
      px3(3)=px+6.0
c
      call xgpl(5,px1,py1)
      call xgpl(3,px2,py2)
      call xgpl(3,px3,py2)
c
      return
      end
```

UNICAMP

```
c ========== subrotina fig4 ==========
c
c finalidade: tracar primitiva agrupamento a partir do ponto
c             fornecido
c
c linguagem: FORTRAN-77
c
c ==========================================================
c
      subroutine fig4(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------
c
      real          px,py,px1(5),py1(5),px2(5),py2(5),px3(5),
     *              px4(5),py4(5),px5(2),py5(2),px6(2)
c
c ---------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+6.0
      py1(2)=py
      px1(3)=px+6.0
      py1(3)=py+4.0
      px1(4)=px
      py1(4)=py+4.0
      px1(5)=px
      py1(5)=py
c
      px2(1)=px+1.0
      py2(1)=py+1.5
      px2(2)=px+3.0
      py2(2)=py+1.5
      px2(3)=px+3.0
      py2(3)=py+3.5
      px2(4)=px+1.0
      py2(4)=py+3.5
      px2(5)=px+1.0
      py2(5)=py+1.5
c
      px3(1)=px+4.0
      px3(2)=px+5.0
      px3(3)=px+5.0
      px3(4)=px+4.0
      px3(5)=px+4.0
c
      px4(1)=px+3.0
      py4(1)=py+1.5
      px4(2)=px+3.5
      py4(2)=py+2.0
      px4(3)=px+3.0
      py4(3)=py+2.5
      px4(4)=px+2.5
      py4(4)=py+2.0
      px4(5)=px+3.0
      py4(5)=py+1.5
c
      px5(1)=px+2.0
      py5(1)=py+2.0
```

```
      px5(2)=px+3.5
      py5(2)=py+2.0
C
      px6(1)=px+3.5
      px6(2)=px+4.0
C
      call xgpl(5,px1,py1)
      call xgpl(5,px2,py2)
      call xgpl(5,px3,py2)
      call xgpl(5,px4,py4)
      call xgpl(2,px5,py5)
      call xgpl(2,px6,py5)
C
      return
      end
```

UNICAMP

```
c ==========  subroutine  fig5  ==========
c
c finalidade: tracar primitiva entidade fresa a partir do
c             ponto fornecido
c
c linguagem: FORTRAN-77
c
c ===========================================================
c
      subroutine fig5(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------------
c
      real          px,py,px1(5),py1(5),px2(5),py2(5)
c
c ------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+5.0
      py1(2)=py
      px1(3)=px+5.0
      py1(3)=py+4.0
      px1(4)=px
      py1(4)=py+4.0
      px1(5)=px
      py1(5)=py
c
      px2(1)=px+1.0
      py2(1)=py+1.0
      px2(2)=px+5.0
      py2(2)=py+1.0
      px2(3)=px+5.0
      py2(3)=py+3.0
      px2(4)=px+1.0
      py2(4)=py+3.0
      px2(5)=px+1.0
      py2(5)=py+1.0
c
      call xgpl(5,px1,py1)
      call xgpl(5,px2,py2)
c
      return
      end
```

UNICAMP

```
c ==========  subrotina  fig6 ==========================
c
c finalidade: tracar primitiva entidade complexa a partir do
c             ponto fornecido
c
c linguagem: FORTRAN-77
c
c ===========================================================
c
      subroutine fig6(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------
c
      real          px,py,px1(5),py1(5),px2(2),py2(2),px3(2)
c
c -------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+6.0
      py1(2)=py
      px1(3)=px+6.0
      py1(3)=py+4.0
      px1(4)=px
      py1(4)=py+4.0
      px1(5)=px
      py1(5)=py
c
      px2(1)=px+1.0
      py2(1)=py
      px2(2)=px+1.0
      py2(2)=py+4.0
c
      px3(1)=px+5.0
      px3(2)=px+5.0
c
      call xgpl(5,px1,py1)
      call xgpl(2,px2,py2)
      call xgpl(2,px3,py2)
c
      return
      end
```

UNICAMP

```
c ==========  subrotina  fig7 ==========================
c
c finalidade: tracar primitiva de ligacao a partir do ponto
c             fornecido
c
c linguagem: FORTRAN-77
c
c ===========================================================
c
      subroutine fig7(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------
c
      real          px,py,px1(2),py1(2)
c
c -------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+6.0
      py1(2)=py+1.5
c
      call xgpl(2,px1,py1)
c
      return
      end
```

```
c  ================================================================
c
c  Nome do arquivo: [illegible]  Gera o arquivo de saída formatado
c
c  Finalidade:
c
c  ================================================================
c
      subroutine [illegible](nx,ny)
      implicit [illegible]
c
c     ------------ Declaração de variáveis ------------------------
c
      real*4     x,y,nx1(3),ny1(3),nx2(3),ny2(3)
c
c     -------------------------------------------------------------
c
      nx1(1)=0.
      ny1(1)=0.
      nx1(2)=0.
      ny1(2)=[illegible]
c
        x1(1)=[illegible]
      ny1(3)=[illegible]
      nx1(1)=[illegible]
      ny1(3)=nx1(3)*
      nx1(3)=[illegible]
      ny1(3)=[illegible]
c
      x1=[illegible]
      y1=[illegible]
c
      [illegible]
      [illegible]
```

```
c ============ subrotina  fig9 ==============================
c
c finalidade: tracar tracar primitiva relacionamento mutuamen-
c             te exclusivo a partir do ponto fornecido
c             fornecido
c
c linguagem: FORTRAN-77
c
c ============================================================
c
      subroutine fig9(px,py)
      implicit none
c
c -------- declaracao de variaveis ---------------------------
c
      real          px,py,px1(5),py1(5),px2(4),py2(4),px3(4)
c
c ------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+3.0
      py1(2)=py+3.0
      px1(3)=px
      py1(3)=py+6.0
      px1(4)=px-3.0
      py1(4)=py+3.0
      px1(5)=px
      py1(5)=py
c
      px2(1)=px-2.0
      py2(1)=py+4.0
      px2(2)=px-1.5
      py2(2)=py+5.5
      px2(3)=px-1.5
      py2(3)=py+2.5
      px2(4)=px-2.0
      py2(4)=py+3.0
c
      px3(1)=px-2.0
      px3(2)=px+1.5
      px3(3)=px+1.5
      px3(4)=px+2.0
c
      call xgpl(5,px1,py1)
      call xgpl(4,px2,py2)
      call xgpl(4,px3,py2)
c
      return
      end
```

UNICAMP

```
c ==========  subrotina  fig10 ==========================
c
c finalidade: tracar primitiva relacionamento fraco essencial
c             a partir do ponto fornecido
c
c linguagem: FORTRAN-77
c
c ================================================================
c
      subroutine fig10(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------------
c
      real           px,py,px1(5),py1(5),px2(2),py2(2),px3(2)
c
c ---------------------------------------------------------------
c
      px1(1)=px
      py1(1)=py
      px1(2)=px+3.0
      py1(2)=py+3.0
      px1(3)=px
      py1(3)=py+6.0
      px1(4)=px-3.0
      py1(4)=py+3.0
      px1(5)=px
      py1(5)=py
c
      px2(1)=px-2.0
      py2(1)=py+3.0
      px2(2)=px-2.0
      py2(2)=py+6.0
c
      px3(1)=px+2.0
      px3(2)=px+2.0
c
      call xgpl(5,px1,py1)
      call xgpl(2,px2,py2)
      call xgpl(2,px3,py2)
c
      return
      end
```

UNICAMP

```
c ==========  subrotina  fig11 ==========
c
c finalidade: tracar primitiva esquema fraco a partir do ponto
c             fornecido
c
c linguagem: FORTRAN-77
c
c ===========================================================
c
      subroutine fig11(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------------
c
      real          px,py,px1(5),py1(5),px2(3),py2(3),px3(3),
     *              px4(5),py4(5)
c
c -------------------------------------------------------------
c
      px4(1)=px
      py4(1)=py
      px4(2)=px+8.0
      py4(2)=py
      px4(3)=px+8.0
      py4(3)=py+6.0
      px4(4)=px
      py4(4)=py+6.0
      px4(5)=px
      py4(5)=py
c
      px1(1)=px+1.0
      py1(1)=py+1.0
      px1(2)=px+7.0
      py1(2)=py+1.0
      px1(3)=px+7.0
      py1(3)=py+5.0
      px1(4)=px+1.0
      py1(4)=py+5.0
      px1(5)=px+1.0
      py1(5)=py+1.0
c
      px2(1)=px+1.0
      py2(1)=py+1.0
      px2(2)=px+2.0
      py2(2)=py+3.0
      px2(3)=px+1.0
      py2(3)=py+5.0
c
      px3(1)=px+7.0
      px3(2)=px+6.0
      px3(3)=px+7.0
c
      call xgsln(1)
      call xgpl(3,px4,py4)
      call xgsln(2)
      call xgpl(5,px1,py1)
      call xgpl(3,px2,py2)
      call xgpl(3,px3,py2)
c
      return
```

end

```
c ==========  subrotina  fig12  ==========
c
c finalidade: tracar primitiva agrupamento fraco a partir do
c             ponto fornecido
c
c linguagem: FORTRAN-77
c
c ==========================================================
c
      subroutine fig12(px,py)
      implicit none
c
c -------- declaracao de variaveis ----------------------
c
      real          px,py,px1(5),py1(5),px2(5),py2(5),px3(5),
     *              px4(5),py4(5),px5(2),py5(2),px6(2),px7(5),
     *              py7(5)
c
c ------------------------------------------------------------
c
      px7(1)=px
      py7(1)=py
      px7(2)=px+3.0
      py7(2)=py
      px7(3)=px+3.0
      py7(3)=py+5.0
      px7(4)=px
      py7(4)=py+5.0
      px7(5)=px
      py7(5)=py
c
      px1(1)=px+1.0
      py1(1)=py+1.0
      px1(2)=px+7.0
      py1(2)=py+1.0
      px1(3)=px+7.0
      py1(3)=py+5.0
      px1(4)=px+1.0
      py1(4)=py+5.0
      px1(5)=px+1.0
      py1(5)=py+1.0
c
      px2(1)=px+2.0
      py2(1)=py+2.5
      px2(2)=px+3.0
      py2(2)=py+2.5
      px2(3)=px+3.0
      py2(3)=py+3.5
      px2(4)=px+2.0
      py2(4)=py+3.5
      px2(5)=px+2.0
      py2(5)=py+2.5
c
      px3(1)=px+3.0
      px3(2)=px+6.0
      px3(3)=px+5.0
      px3(4)=px+5.0
      px3(5)=px+6.0
c
      px4(1)=px+4.0
```

```
      py4(1)=py+2.5
      px4(2)=px+4.5
      py4(2)=py+3.0
      px4(3)=px+4.0
      py4(3)=py+3.5
      px4(4)=px+3.5
      py4(4)=py+3.0
      px4(5)=px+4.0
      py4(5)=py+2.5
C
      px5(1)=px+3.0
      py5(1)=py+3.0
      px5(2)=px+3.5
      py5(2)=py+3.0
C
      px5(1)=px+4.5
      px5(2)=px+5.0
C
      call xgsln(1)
      call xgpl(5,px7,py7)
      call xgsln(1)
      call xgpl(5,px1,py1)
      call xgpl(5,px2,py2)
      call xgpl(5,px3,py2)
      call xgpl(5,px4,py4)
      call xgpl(2,px5,py5)
      call xgpl(2,px6,py5)
C
      return
      end
```

```
c*****common /pontos/ ********************************
c
      common /pontos/   posegx(24),posegy(24),segtip(24),
     *                  segcor,FIRSEG
c
      integer           segtip,FIRSEG,segcor
c
      real              posegx,posegy
```