Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Comunicações

Procedimentos para o Desenvolvimento de Ferramentas (Software) com Aplicações em Processamento Digital de Imagem

Alexandra Maria Rodrigues Cardoso

Orientador :
Prof. Dr. Yuzo Iano

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do Título de Mestre em Engenharia Elétrica.

Campinas, SP - Brasil
1998

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C179p

Cardoso, Alexandra Maria Rodrigues
Procedimentos para o desenvolvimento de ferramentas (software) com aplicações em processamento digital de imagem. / Alexandra Maria Rodrigues Cardoso.--Campinas, SP: [s.n.], 1998.

Orientador: Yuzo Iano
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento de imagens – Técnicas digitais . 2. Programação visual (Computação). 3. Televisão digital. I. Iano, Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RESUMO

O objetivo desta dissertação é sugerir procedimentos que permitam o aproveitamento de um ambiente já existente para se realizar simulações com programação visual tendo como aplicação específica o processamento digital de sinais de televisão convencional e avançada. O ambiente Khoros possui recursos para o Processamento Digital de Sinais (PDS), sendo que a função primordial do processamento digital aplicado à televisão é a de fornecer ferramentas para facilitar a identificação e a extração da informação contidas nas imagens. Nossa contribuição é adicionar funções específicas no ambiente Khoros estabelecendo novas possibilidades quanto à programação necessária para implementação de modelos de simulação para sistemas de televisão digital.

ABSTRACT

The objective of this dissertation is to suggest procedures that allow the utilization of an existing environment to carry out simulations using a visual programming language in applications such as Digital Image Processing(DIP) in Conventional Television(CTV) and Advanced Television(ATV). That is done by introducing additional tools to the Khoros Digital Signals Processing resources, since the major function of Digital Signal Processing applied to the television is to provide tools to facilitate the identification and the extraction of the information contained into images. Our contribution is to add specific functions to the Khoros environment, stabilishing new possibilities of programming, necessary for simulation model implementations for digital television systems.

Agradecimentos

Agradeço especialmente ao meu orientador, Prof. Dr. Yuzo Iano, pelo apoio e orientação recebidos no dia a dia.

Aos Professores Álvaro Luiz Stelle e Edson Moschin pelo tempo e contribuição dedicados a este trabalho.

Aos Professores João B. T. Yabu-uti, Roberto Lotufo, Dalton Soares Arantes, Clayton Bezzan pela colaboração e contribuição a este trabalho.

Aos Professores Jocélio Souza de Sá, Rodrigo G. Santana, José Antônio Justino Ribeiro e Ely Kallás do Instituto Nacional de Telecomunicações - INATEL, de Santa Rita do Sapucaí, pela atenção e pelo incentivo recebido.

Ao colega Ayres M. A. do Nascimento pelas rotinas de conversão de formatos de sinais de TV e ATV cedidas para este trabalho e também pela amizade e incentivo recebidos. Ao colega Ricardo Massahiro Nishihara pelo tempo dedicado a este trabalho com orientações sobre o Khoros na versão 1.0.

Aos colegas e amigos Mylène C. Q. de Farias, Marcelo M.de Carvalho, Maria Luiza de Moraes Melo, Antonio Moraes, Antonio Fançony, Guillermo Kemper Vasquez, Leone Correa, Vicente Becerra, Evaldo Gonçalves Pelaes, Edmilson da Silva Moraes, Luiz Rômulo Mendes e Rene Togni del Pietro pela amizade e incentivo.

Aos funcionários da FEEC pelo auxílio e em particular à secretária Maria Lúcia Costa Cardoso pela prestativa atenção.

Agradeço fortemente à Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP, pelo suporte financeiro a este trabalho.

Aos meus pais Amadeu R. Patrocínio e Maria Suzana F. Rodrigues e aos meus irmãos Adriana A. Rodrigues e Anderson M. Rodrigues pelo carinho e incentivo recebidos.

Agradeço muitíssimo ao meu marido Fabbryccio Akkazzha C. M. Cardoso pelo grande apoio e valiosas contribuições a este trabalho. Também sou grata aos momentos felizes de alegria, carinho e descontração, porque sem esses preciosos momentos já não sei mais viver.

Finalmente agradeço a Deus, meu pai do céu, que me guia e me sustenta no caminho do amor.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1. Considerações Iniciais

Nos últimos anos o campo do processamento e transmissão digital vem crescendo consideravelmente. Nas mais variadas aplicações, o processamento digital de vídeo vem sendo objeto de estudo em vários centros de pesquisa no mundo todo; visando os mais variados serviços e utilizações. Por conseguinte, os serviços para televisão tem encontrado novos campos através da digitalização dos sinais de vídeo.

Nas mais diversas aplicações e ramos de utilização, principalmente com o surgimento da televisão de alta definição, a televisão digital vem conseguindo conquistar um espaço cada vez maior, tendo portanto um futuro bastante promissor. A função primordial do processamento digital aplicado à televisão é a de fornecer ferramentas para facilitar a identificação e a extração da informação contidas nas imagens. Nesse sentido, sistemas dedicados de computação são utilizados para atividades interativas de análise e manipulação das imagens.

O Laboratório de Comunicações Visuais do grupo PDI/DECOM/UNICAMP, necessita de um ambiente onde seja possível a implementação de modelos de simulação para esquemas envolvendo sistemas de televisão digital. Este trabalho vem propor um ambiente de domínio público já existente, o Khoros, para tal função. Para a realização destas simulações é necessário a inserção dos programas desenvolvidos pelo grupo PDI/DECOM/UNICAMP no ambiente Khoros. Então, são sugeridos procedimentos para a inserção destes programas. O procedimento desenvolvido é útil para todos aqueles que tem problemas similares

de implantação de um ambiente de simulação para processamento de sinais. Portanto, o trabalho realizado é importante como uma proposta para ambientes que utilizam o Khoros entre suas ferramentas.

1.2. Sobre o Khoros

O Khoros é um ambiente de integração e desenvolvimento de software que enfatiza o processamento da informação e vizualização de dados. O ambiente Khoros foi desenvolvido pelo "The Khoros Group, Departament of Electrical and Computer Engineering, University of New Mexico, Albuquerque"[1].

O Khoros é também um software de domínio público para pesquisa em processamento e visualização digital de sinais baseado no sistema X11R4[2][3]. Consiste de uma infraestrutura baseada em vários sistemas de camadas interativas.

O pacote é composto de :

- linguagem de programação visual;
- geradores de códigos para extensão da linguagem visual e adição de novos pacotes de aplicação para sistema;
- uma interface para edição interativa;
- um pacote interativo de visualização de imagens;
- uma extensa biblioteca para processamento de sinais, análises numéricas, rotinas de processamento de sinais;
- pacotes de visualização gráfica 2D/3D.

Aplicações em X Windows :

- Animate : ferramenta interativa de visualização de sequências de imagens;
- Cantata : linguagem de programação visual;
- Concert : sistema para distribuição de interfaces X de usuários;
- Editimage : ferramenta de manipulação e visualização interativa de imagens;
- Xprism2/3 : pacotes de visualização gráfica 2D e 3D;
- e outros.

O Khoros contem mais de 260 programas classificados nas seguintes categorias :

- aritmética ;
- estatística ;
- classificação ;
- conversão de cor ;
- conversão de dados ;
- conversão de formatos ;
- extração de características ;
- filtros no domínio da freqüência ;
- álgebra matricial ;
- filtros no domínio espacial ;
- manipulação geométrica ;
- manipulação de histograma ;
- geração de sinais ;
- operações lineares ;
- segmentação ;
- estimação espectral ;
- transformadas ;
- subregiões.

1.3. Organização da tese

Este trabalho se resume em cinco capítulos :

O Capítulo 2 tem como objetivo apresentar uma introdução ao Khoros, um ambiente de domínio público de integração e desenvolvimento de software que enfatiza o processamento da informação e visualização de dados. Todos os programas de visualização e processamento da informação do Khoros estão disponíveis via linguagem de programação visual, Cantata. Então faz-se uma introdução ao Cantata, uma linguagem de programação

visual, apresentando vários elementos componentes da ferramenta. Este capítulo tende a elucidar a ferramenta de programação visual, o Cantata, mostrando as facilidades de uso da mesma.

No Capítulo 3, são apresentadas as ferramentas de desenvolvimento de programas : Craftsman, Composer e Guise.

O Capítulo 4 descreve os procedimentos utilizados para a inserção de um programa no ambiente Khoros. Neste capítulo explica-se o propósito de cada arquivo associado com cada tipo de programa, assim como várias questões relacionadas ao desenvolvimento de código usando o Khoros. São também mostrados exemplos simples para alguns dos tipos de programas.

O Capítulo 5 é uma conclusão do trabalho apresentado.

Apêndices dos programas são anexados no final do trabalho.

Finalmente, deixa-se uma ressalva quanto aos termos técnicos usados, uma vez que esses termos são usados em inglês em nosso ambiente de trabalho. A maioria desses termos estão sem aspas e assim mantem-se a nomenclatura original a fim de evitar confusões.

Capítulo 2

Sistema Khoros

2.1. Introdução

O Khoros é um ambiente de integração e desenvolvimento de software que enfatiza o processamento da informação e visualização de dados[5]. Todos os programas de visualização e processamento da informação do Khoros estão disponíveis via linguagem de programação visual, cantata.

Um programa em linguagem C para o ambiente Khoros pode ser criado utilizando um conjunto de ferramentas fornecidas pelo Khoros. É simples e direto usando as ferramentas Craftsman, Composer e Guise do próprio Khoros. Usando as ferramentas do Khoros torna-se mais fácil o desenvolvimento de programas e garante-se que todos os programas possam usar a biblioteca Khoros para manipulação de dados.

2.1.1. O ambiente Khoros

O sistema Khoros está dividido entre várias toolboxes. Uma toolbox é uma coleção de programas e/ou bibliotecas que possuem funções similares ou um objetivo comum. As toolboxes que são distribuidas com o Khoros são :

A toolbox bootstrap

Contém a base necessária para a instalação do Khoros. Contém bibliotecas que fazem os serviços básicos, matemáticos, etc.

A toolbox dataserv

Contém bibliotecas que fazem serviços de dados.

A toolbox design

Contém as aplicações : Cantata(linguagem de programação visual), Craftsman(programa de gerência de toolbox), Composer(editor de "software object"), Guise(ferramenta de manipulação da interface para o usuário) e outras.

A toolbox envision

Fornece aplicações de visualização de dados. As principais aplicações são : animate(um programa de animação de imagens), editimage(ferramenta de manipulação e visualização da imagem), editcmap(ferramenta para modificação de colormaps), extractor(extração de uma região de interesse da imagem), getimage(extrai imagens da tela da estação de trabalho), putdata(ferramenta não interativa para visualização de dados), xprism(pacote 2D/3D), spectrum(programa interativo para a classificação da imagem).

A toolbox datamanip

Contém operadores de manipulação de dados.

A toolbox support

É um depósito para vários programas do Khoros.

A toolbox image

Contém operadores de processamento de imagem.

A toolbox migration

Fornece ferramentas e documentação para ajudar a conversão de programas do Khoros 1 para a versão atual.

A toolbox geometry

Contém rotinas para o processamento de dados para visualização 3D.

A toolbox sampledata

Contém amostras de dados que podem ser usados no sistema Khoros.

2.1.2. dependências entre as toolboxes

Todas as toolboxes do Khoros dependem da toolbox Bootstrap. A toolbox Dataserv só depende da toolbox Bootstrap, devendo ser instalada após a toolbox Bootstrap. As toolboxes Design, Geometry, Migration e Datamanip dependem da Dataserv. As toolboxes Support e Envision dependem da toolbox Design, enquanto as toolboxes Image e Matrix dependem da toolbox Datamanip. A toolbox Sampledata não depende de nenhuma outra toolbox, porque ela contém somente amostras de dados. Na Figura 2.1, pode-se ver de maneira gráfica a dependência entre as toolboxes.

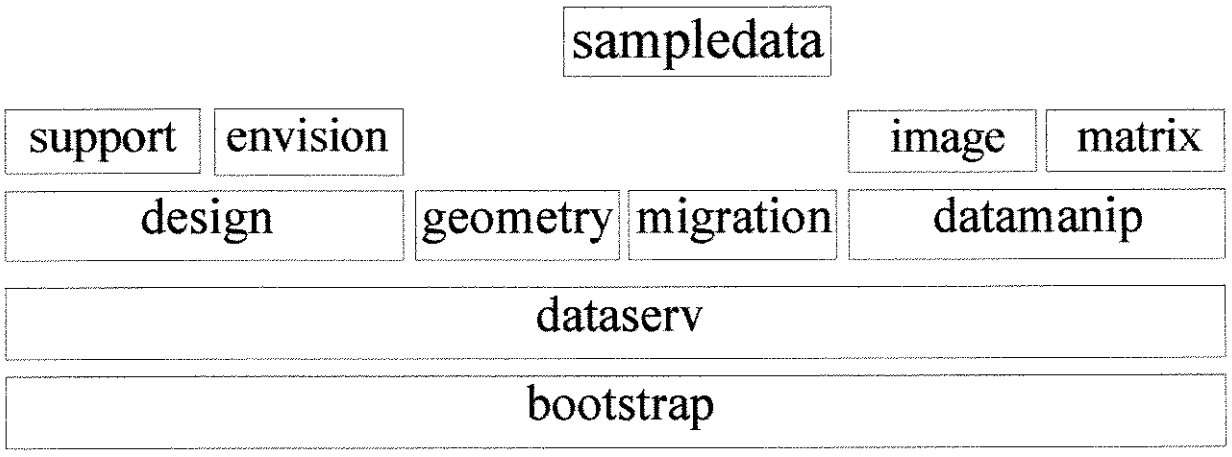

Figura 2.1: Dependência entre as toolboxes do sistema Khoros.

2.1.3. Ferramentas do Khoros

Cantata

A linguagem de programação visual é a principal aplicação oferecida pela toolbox Design. O Cantata é um ambiente de programação visual oferecido pelo sistema Khoros.

Craftsman

É uma ferramenta para gerência de toolbox, oferecida pela toolbox Design. É usada para criar, apagar e copiar toolbox assim como bibliotecas e programas.

Composer

Também distribuido pela toolbox Design, o Composer é um editor de programas (software object). Trabalha em conjunto com o Craftsman.

Guise

Ferramenta de criação e manipulação direta da interface gráfica para o usuário (GUI-Graphical User Interface), que trabalha em conjunto com o Composer. É fornecida pela toolbox Design.

Kman

Parte da toolbox Bootstrap, este comando permite o acesso às páginas man de qualquer programa do Khoros.

Editimage

A toolbox Envision fornece a ferramenta Editimage. Esta é uma ferramenta de visualização, manipulação e exame da imagem.

Xprism

Pacote 2D/3D oferecido pela toolbox Envision.

Animate

Parte da toolbox Envision, o programa Animate faz animações interativas de seqüências da imagem.

Spectrum

Também da toolbox Envision, Spectrum é um sistema interativo de classificação do sinal/imagem.

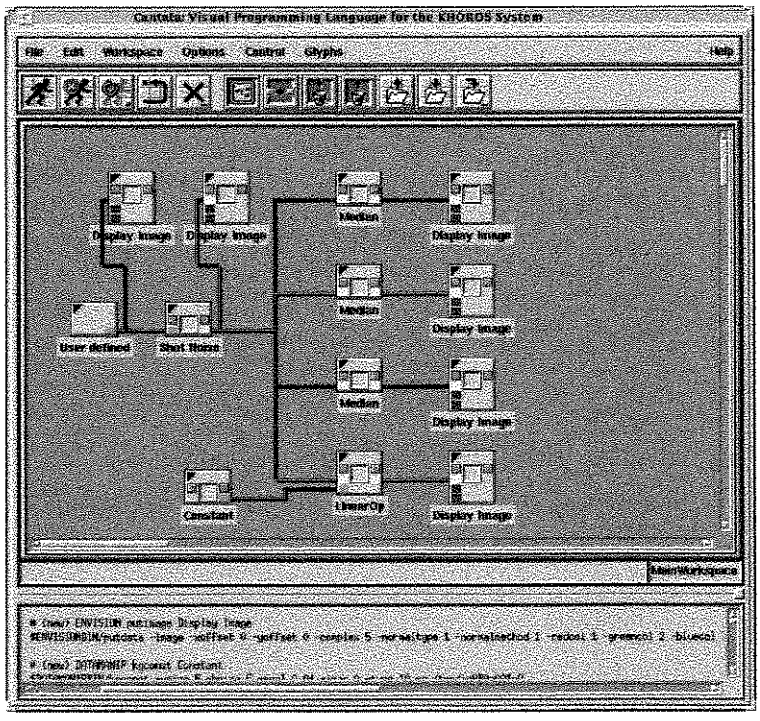

Figura 2.2: Ilustração de um programa visual no workspace do Cantata.

2.2. Cantata

Cantata : Ilustrado na Figura 2.2, é um ambiente de programação visual para o sistema Khoros. Ele é usado para criar programas visuais e pode também ser usado para encapsular um programa visual, criando assim um ícone para tal programa. A programação visual é semelhante ao diagrama em blocos. No Cantata, os ícones (chamados de glyphs) representam programas do Khoros. No entanto, sabendo que o Khoros é um ambiente de integração de programas, os glyphs podem também representar programas que foram inseridos no Khoros. Para criar um programa visual, selecionam-se os programas desejados, colocando-se estes glyphs no ambiente de trabalho (chamado workspace) do cantata, e conecta-se os mesmos entre si, formando uma rede. Os programas visuais podem ser executados, salvos e restituídos para serem usados novamente ou modificados mais tarde.

2.2.1. Inicializando o Cantata

Para inicializar o cantata, deve-se primeiro rodar o OpenWindows. O Cantata pode então ser inicializado pela janela do Command Tool. Escreva, depois do prompt, source /usr/khoros2.2/khoros_env e pressione ENTER. Escreva cantata depois do prompt e pressione ENTER. Para finalizar o Cantata quando em operação, escolha a opção "Exit" do menu "File", localizado no canto superior esquerdo na janela do Cantata.

2.2.2. Composição da janela do Cantata

A interface gráfica para o usuário (GUI-Graphical User Interface) é uma janela que possui um menu principal e uma área de trabalho (workspace) com uma barra de comandos. No workspace do Cantata pode ser criado uma rede de glyphs. A barra de comandos fornece fácil acesso às opções mais usadas. O menu principal do cantata contem menus com as opções conforme a Tabela 2.1.

O menu "File"

Este menu fornece ferramentas para manipular arquivos de workspace. Usando este menu, pode-se salvar e restaurar arquivos de workspace.

- New - cria um workspace novo
- Open... - abre um arquivo browser para localizar e abrir arquivos de workspace.
- Save - salva o atual workspace
- Save As... - permite que o atual workspace seja salvo com um novo nome.
- Close - fecha o atual workspace
- Exit - sai do cantata

O menu "Edit"

Com este menu pode-se copiar, apagar e editar programas visuais. Também fornece ferramentas para localizar glyphs.

Menu principal - Cantata						
File	**Edit**	**Workspace**	**Options**	**Control**	**Glyphs**	**Help**
New	Cut	Run	Preferences	Create Procedure	categorias, subcategorias, glyphs	On Cantata
Open	Copy	Stop	Hide Command Bar	Create Count Loop		Activate Tooltips
Save	Paste	Reset	Hide Console	Create While Loop		License
Save As...	Show Clipboard	Run Once	Small Size	Expression		
Close	Duplicate	Clear	Clear Console	If/Else		
Exit	Delete	Redraw		Merge Paths		
	Undo Delete	Information		Switch		
	Select All	Variables...		Trigger		
	Unselect All	Configure Remote Hosts...				
	Find					

Tabela 2.1: Controle do menu principal do Cantata

- Cut - remove um glyph selecionado do workspace e coloca-o no clipboard
- Copy - faz uma cópia do glyph selecionado e coloca-a no clipboard
- Paste - coloca no workspace o glyph do clipboard
- Show Clipboard - mostra o clipboard. O clipboard é usado como um "depósito" temporário.
- Duplicate - faz a cópia do glyph selecionado.
- Delete - remove o glyph selecionado.
- Undo Delete - retorna um glyph que foi apagado.
- Select All - seleciona todos os glyphs e conexões do workspace.
- Unselect All - desfaz a selecão dos itens do workspace que foram selecionados.
- Find - Mostra uma janela Routines/Finder. Esta janela é utilizada para procurar glyphs disponíveis do menu Glyphs.

O menu "Workspace"

Fornece as opções de executar, parar e manipular programas visuais no workspace.

- Run - é usado para executar o programa visual desde que os glyphs estejam selecionados e conectados.
- Stop - para a execução do programa.
- Reset - é usado para resetar todos os glyphs do workspace
- Run Once - executa o programa uma única vez
- Clear - apaga todos os glyphs do workspace
- Redraw - atualiza todos os glyphs e conexões da rede no workspace.
- Information - Toma algumas informações do programa visual como o número de glyphs e outras.
- Variables... - Mostra a janela Variables. Esta janela pode ser usada para definir novas variáveis, ver os valores de todas as variáves e vê-las como são incrementadas durante a execução do programa.
- Configure Remote Hosts... - Cantata permite que programas sejam executados em várias máquinas.

O menu "Options"

Fornece os seguintes itens :

- Preferences... - permite mudar as características do workspace. Por exemplo : o tamanho, a cor da grade, a textura e a cor do workpace e outros.
- Hide Command Bar - Faz a barra de comandos aparecer ou desaparecer.
- Hide Console - Faz o console desaparecer
- Small Size - Diminui o tamanho da janela do cantata
- Clear Console - Apaga toda mensagem que está no console.

O menu "Control"

Fornece uma série de utilidades para criação de estruturas especiais no workspace do Cantata.

- Create Procedure - cria procedimentos
- Create Count Loop - cria um contador de loop
- Create While Loop - cria um loop while
- Expression - fixa os valores das variáveis.
- If/Else - direciona o fluxo de dados para um caminho se a condição for satisfeita e para outro caminho se a condição não for satisfeita.
- Merge Paths - direciona o fluxo de dados de dois caminhos separados para o mesmo caminho.
- Switch - seleciona uma de duas entradas para ser usada pela rede visual, dependendo do valor da declaração original.
- Trigger - atrasa a execução de um glyph até que os dados estejam disponíveis para outro glyph.

O menu "Glyphs"

Todas as rotinas disponíveis como glyphs no Cantata são acessíveis através do menu Glyphs. No menu Glyphs, encontra-se uma lista de categorias. Cada categoria possui

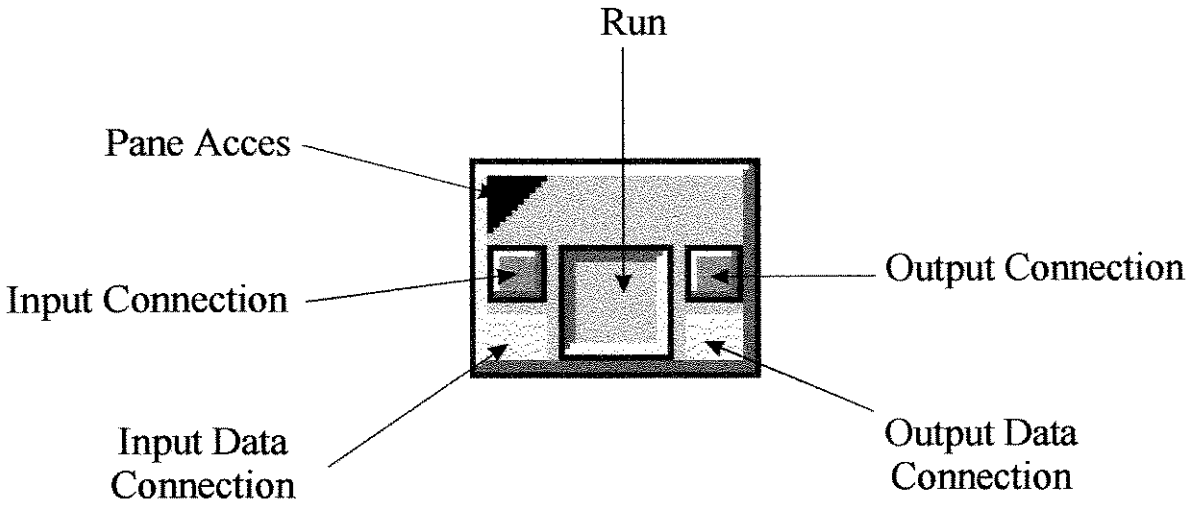

Figura 2.3: Glyph : representação visual de um programa disponível no Cantata.

uma lista de subcategorias. Cada subcategoria apresenta uma lista de glyphs. Quando um desses nomes de glyphs é selecionado, ele irá aparecer no workspace do Cantata.

O menu "Help"

Fornece informação sobre o Cantata

- On Cantata - informações sobre o Cantata
- Active Tooltips - mostra uma breve descrição quando o cursor do mouse passa pela barra de comandos, glyphs, conexões, etc.
- License - fornece informação sobre a licença do Khoros.

2.2.3. Glyph

Um glyph é simplesmente uma representação visual de um programa disponível no Cantata, como ilustrado na Figura 2.3.

Componentes de um glyph

As componentes de um glyph são :

- Input Data Connection Node - Cada glyph pode ter uma ou mais entradas de dados.
- Output Data Connection Node - Pode-se ter uma ou mais saídas de dados.
- Pane Access Button - Todo glyph apresenta, no canto superior esquerdo, um botão na forma de um triângulo. Este botão é usado para acessar o pane do glyph. No pane especificam-se valores para argumentos do operador. Estes argumentos são os mesmos da linha de comando quando o operador está executando este programa fora do Cantata.
- Run button - O botão que fica no centro do glyph é usado para executar o programa. Alguns glyphs não possuem este botão, são aqueles que fornecem entrada para outros glyphs.
- Input Control Connection node - É usado para atrasar a execução do glyph até que o outro seja executado.
- Output Control Connection Node - É usado para atrasar a execução de outro glyph até que este glyph seja executado.
- Operator Name - nome do operador que está representando.

Operações Básicas

- Mover - Depois de selecionado, com o botão esquerdo do mouse, arraste o glyph até o local desejado.
- Destruir - Para eliminar um glyph do workspace, selecione o glyph a ser destruido e o botão "Delete Select Glyphs" na barra de comandos.
- Executar - Basta pressionar o botão que fica no centro do glyph.
- Mudar o nome do Glyph - Selecione o atual nome do glyph, que fica logo embaixo dele, para aparecer um prompt. Neste prompt deverá ser colocado o novo nome para o glyph.

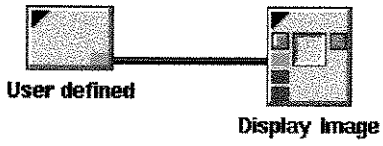

Figura 2.4: Exemplo de uma conexão de dado.

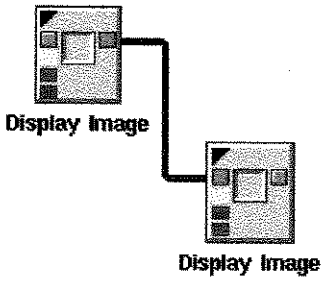

Figura 2.5: Exemplo de uma conexão de controle.

2.2.4. Conexões entre glyphs

Conexões de dados

Os glyphs apresentam entrada e saída de dados, representados por pequenos quadrados localizados no lado esquerdo e direito respectivamente. Para a conexão entre dois glyphs basta clicar o mouse na saída de dados de primeiro glyph e depois na entrada do segundo glyph(ou vice-versa). Feito isto, uma linha de conexão será desenhada entre os dois glyphs como ilustrado na Figura 2.4.

Conexões de Controle

Para criar uma conexão de controle entre dois glyphs, clica-se o mouse na saída de controle do primeiro glyph e então na entrada de controle do segundo glyph. Quando uma bem sucedida conexão é feita, uma linha de conexão será desenhada entre os dois glyphs como ilustrado na Figura 2.5. Agora o segundo glyph será controlado pelo primeiro.

Manipulando as Conexões

A conexão feita entre dois glyphs pode ser apagada ou pode-se fazer a conexão de um dos glyphs para outro.

Execução

Depois do programa visual contruído, pode-se executar os glyphs de duas maneiras :

1. Executar todo o programa visual de uma só vez.
2. Executar cada glyph individualmente.

Argumentos do programa

Todo programa no sistema Khoros possui uma interface gráfica para o usuário, que é usada com o Cantata. Este GUI possui um item correspondente a cada argumento do programa de modo que o GUI pode ser usado para especificar o valor desejado para cada argumento.

2.2.5. Preferências

A janela de preferências está disponível no menu "Options". Parâmetros que controlam vários aspectos do ambiente de programação visual (Cantata) são estabelecidos por esta janela.

Canvas Grid

A Figura 2.6 mostra a janela(subform) "Preferences" com o botão "Canvas Grid" selecionado.

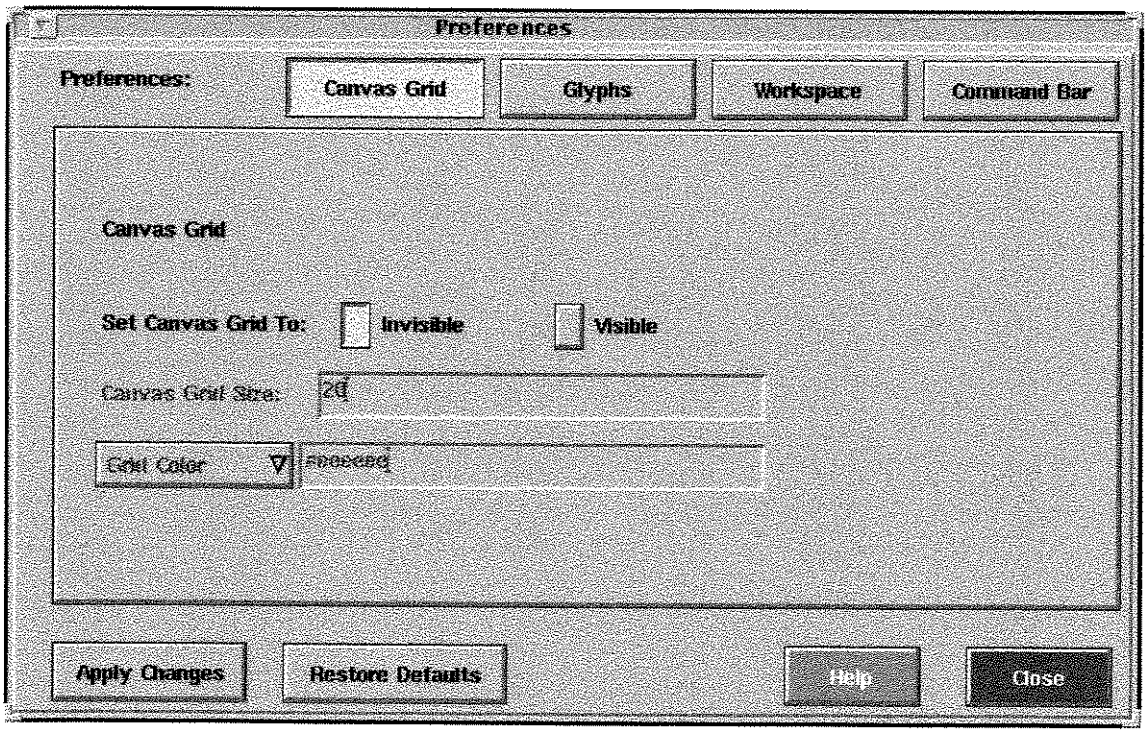

Figura 2.6: Janela(subform) "Preferences" com o botão "Canvas Grid" selecionado.

Glyphs

A Figura 2.7 mostra a janela(subform) "Preferences" com o botão "Glyphs" selecionado.

Workspace

A Figura 2.8 mostra a janela(subform) "Preferences" com o botão "Workspace" selecionado.

Command Bar

A Figura 2.9 mostra a janela(subform) "Preferences" com o botão "Command Bar" selecionado.

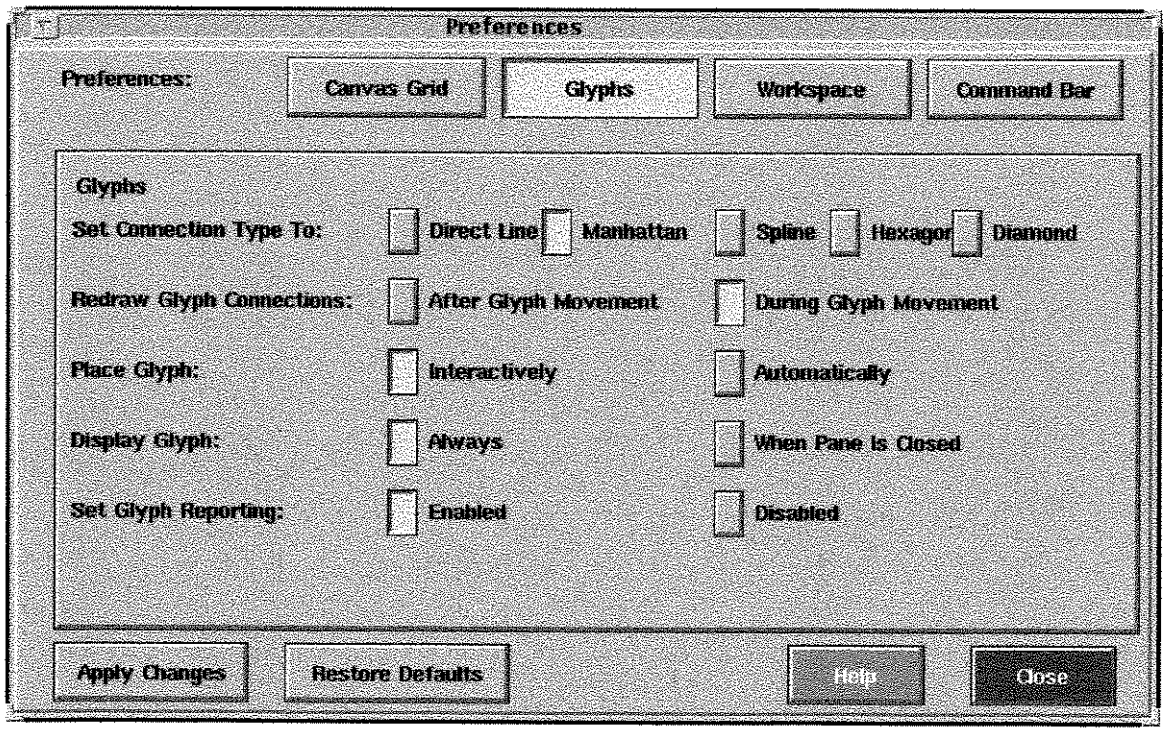

Figura 2.7: Janela(subform) "Preferences" com o botão "Glyphs" selecionado.

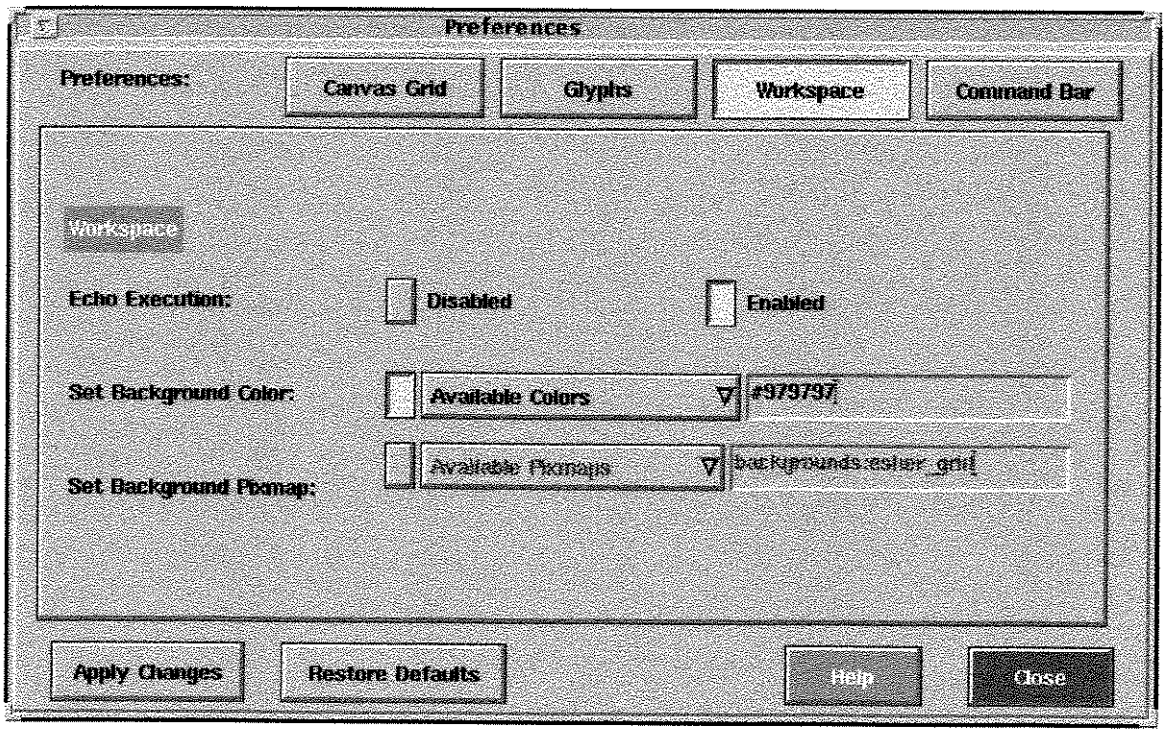

Figura 2.8: Janela(subform) "Preferences" com o botão "Workspace" selecionado.

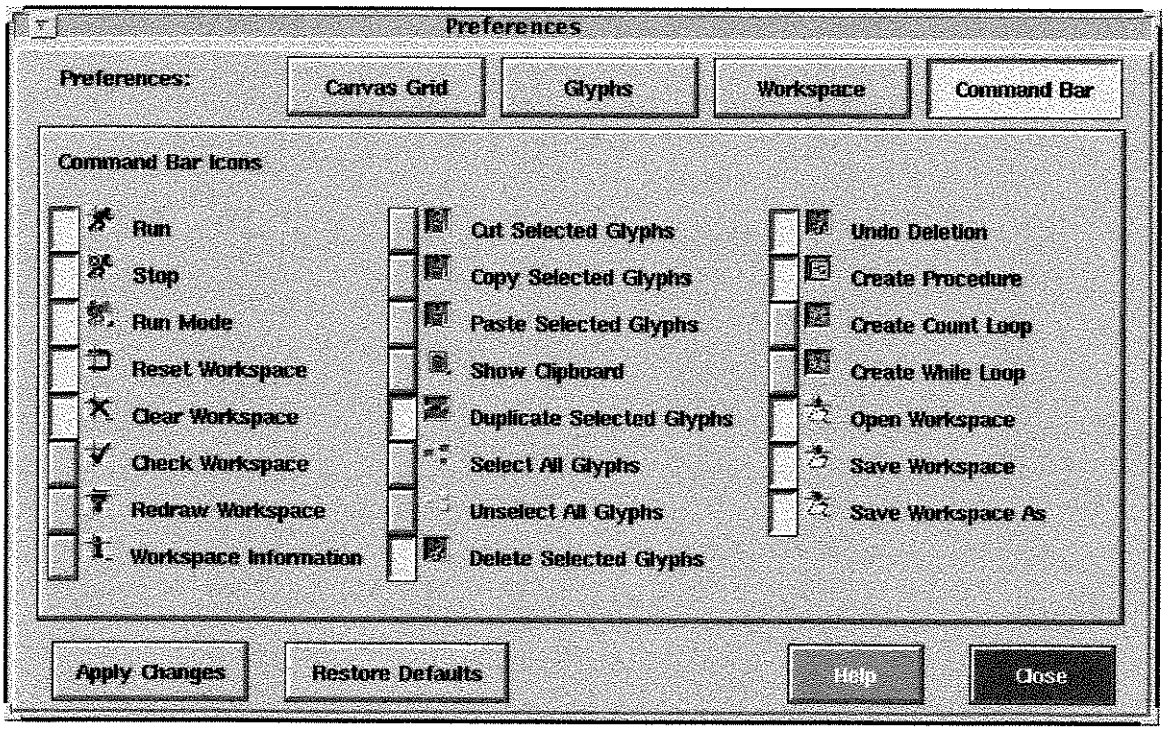

Figura 2.9: Janela(subform) "Preferences" com o botão "Command Bar" selecionado.

2.2.6. Variáveis

A janela(subform) "Variables" é usada para definir variáveis e expressões dentro do ambiente de programação visual - Cantata.

2.2.7. Encapsular workspaces

Uma vez que o programa visual esteja completo, pode-se criar uma aplicação independente que pode ser executada sem que a rede de glyphs, definida como um programa visual, seja mostrada.

2.3. Conclusão

Este capítulo apresenta uma introdução ao sistema Khoros apresentando a linguagem de programação visual, o Cantata. As informaçoes deste capítulo não substituem as dos

manuais do Khoros. Para maiores informações sobre o Cantata recomenda-se a leitura dos manuais do Khoros[5].

Capítulo 3

Desenvolvimento de programas

3.1. Introdução

É aconselhável que seja criada uma toolbox para as diferentes características de programas. Cada programa e biblioteca no sistema Khoros [4] está contido numa toolbox. Por exemplo, programas e bibliotecas de processamento digital de imagens estão contidos na Toolbox Image, já os programas e bibliotecas que fazem operações com matrizes estão na Toolbox Matrix. Uma toolbox é um conjunto de programas e bibliotecas que compartilham algo em comum. Existem cinco categorias associadas aos programas e bibliotecas : kroutine, xvroutine, library, pane e script.

O ambiente de desenvolvimento de programas apresenta duas ferramentas : Craftsman e Composer. Para a gerência de toolbox e programas associados a toolbox. O Craftsman é usado para criar, deletar e copiar toolbox.

A toolbox cumpre uma estrutura de diretório definida na qual os programas são alocados. A Figura 4.1 mostra a estrutura de uma toolbox. Mas quando uma toolbox é criada, ela não apresenta todos esses diretórios. Estes diretórios não são criados até que sejam necessários. Uma toolbox contem os seguintes diretórios :

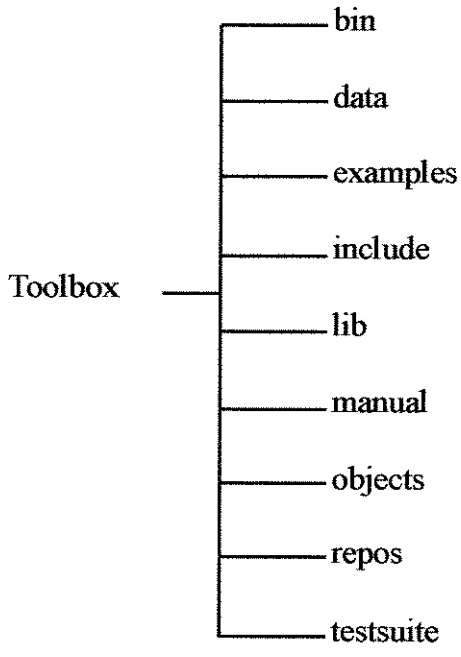

Figura 3.1: A estrutura de diretórios de uma toolbox.

bin	o diretório onde todos os programas executáveis são colocados.
data	possui arquivos de dados que são usados pelos programas desta toolbox.
examples	este diretório contem exemplos de programas.
include	contem o arquivo "toolbox.h" e um subdiretório para cada biblioteca da toolbox.
lib	contem arquivo compilados das bibliotecas da toolbox.
manual	contem um manual para a toolbox.
objects	contem programas que podem ser kroutine, xvroutine, library, pane e script.
repos	um depósito para vários arquivos. Como exemplo, os arquivos de configuração.
testsuite	é um lugar para os vários "test suites" que são criados.

3.2. Programas

Um programa (software object) está contido dentro de uma toolbox e é composto de código fonte, documentação e interface para o usuário. Todos os programas (software objects) possuem arquivos para salvar várias informações, como o nome, a localização e outras. Existem categorias associadas aos programas (software objects) e bibliotecas :

program objects (como as kroutines e xvroutines)
library objects
pane objects
script objects

3.2.1. Program Object

Program Obejcts são divididos em dois tipos : kroutines e xvroutines. Um program object do Khoros consiste de arquivos associados com um programa em particular.

kroutine objects	a maioria dos programas no sistema Khoros são kroutines. Estas incluem todos os programas de processamento de dados, tal como as rotinas de processamento de imagem e sinal.
xvroutine objects	é um programa que não pode ser rodado sem que uma interface gráfica para o usuário seja mostrada. Existem dois tipos de xvroutines : programas interativos e não interativos. Exemplos de programas interativos : xprism, editimage, cantata, guise, craftsman e spectrum. Exemplos de programas não interativo : putdata.

3.2.2. Library Objects

Representa uma coleção de funções que são usadas pelo programa fonte.

3.2.3. Pane Objects

São interfaces para os programas existentes.

3.2.4. Script Objects

Fazem algumas manipulações de programas, mas não possuem funções para manipulação de dados.

3.3. Craftsman

O Craftsman é uma ferramenta para gerenciar toolbox e programas (software objects). A Figura 3.2 mostra a janela do Craftsman, a qual possui duas listas. A lista da esquerda contem as toolboxes disponíveis. A lista da direita mostra todos os programas (objects) da toolbox selecionada. A janela tambem possui o botão "Toolbox Operations" localizado na parte inferior esquerda que fornece as operações básicas com as toolboxes : criar, apagar e mudar atributos. E o botão "Object Operations" localizado na parte inferior direita que fornece as operações básicas para os programas (objects) : criar, apagar, editar, copiar e mudar atributos.

3.3.1. O botão "Toolbox Operations"

Criando uma toolbox

O primeiro item do botão "Toolbox Operations" é o "Create Toolbox". Selecionando este item aparecerá uma janela onde se pode entrar com os atributos da nova toolbox. Deve-se fornecer o nome e o path da nova toolbox. Os atributos restantes podem ser inseridos usando a janela do item "Toolbox Attributes".

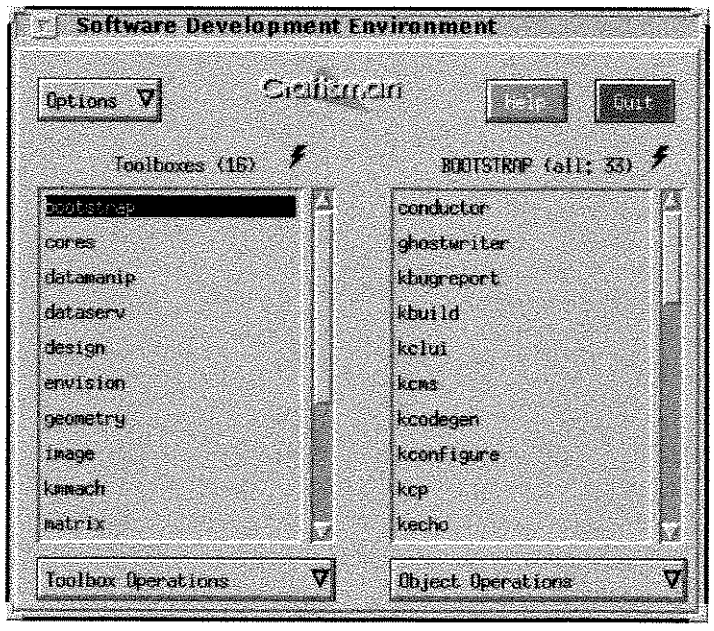

Figura 3.2: Craftsman : ferramenta para gerenciar toolbox e programas.

Atributos de uma toolbox

Toda toolbox possui um número de atributos associados a ela. Estes atributos fornecem informações sobre a toolbox e seu autor. Estes atributos são fornecidos quando a toolbox é criada, e a maioria podem ser mudados. O item "Toolbox Attributes", assim que selecionado, traz uma janela para a modificação de atributos da toolbox.

Criando um manual para a toolbox

Se a toolbox criada será usada por outras pessoas, é necessário a criação de um manual que forneça informações adicionais. Selecionando o item "Create Toolbox Manual" resulta num prompt para confirmação, informando o nome do manual e se deseja continuar. Se continuar, deverá ser criado uma introdução, um glossário e um índice.

Apagando uma toolbox

O último item do menu "Toolbox Operations" é "Delete Toolbox". Se o item "Delete Toolbox" é selecionado, a toolbox selecionada será apagada. Mas antes de remover aparece

um prompt para confirmação. Se confirmado a toolbox será removida e a lista de toolboxes disponíveis será atualizada.

3.3.2. O menu "Object Operations"

Criando um programa

Selecionando o item "Create Objects", aparece uma janela para a criação de programas para a toolbox especificada. Nesta janela, deve-se selecionar o tipo de programa a ser criado. Os programas podem ser do tipo : kroutine, xvroutine, pane, library e script.

Editando um programa

Depois de criado o programa (object) para a toolbox especificada, pode-se editar este programa (object) usando o Composer. Para usar o Composer, basta selecionar a opção "Edit Object (Composer)" e o Composer será executado.

Atributos de um programa

Depois de selecionado um programa da lista da direita, o item "Object Attributes" no menu "Object Opertions" estará ativo. Depois de selecionado, aparecerá uma janela onde serão feitas as alterações dos atributos deste programa (object).

Copiando um programa

Copia um programa (object) já existente no Khoros para a toolbox desejada.

Apagando um programa

Apaga o programa (object) selecionado.

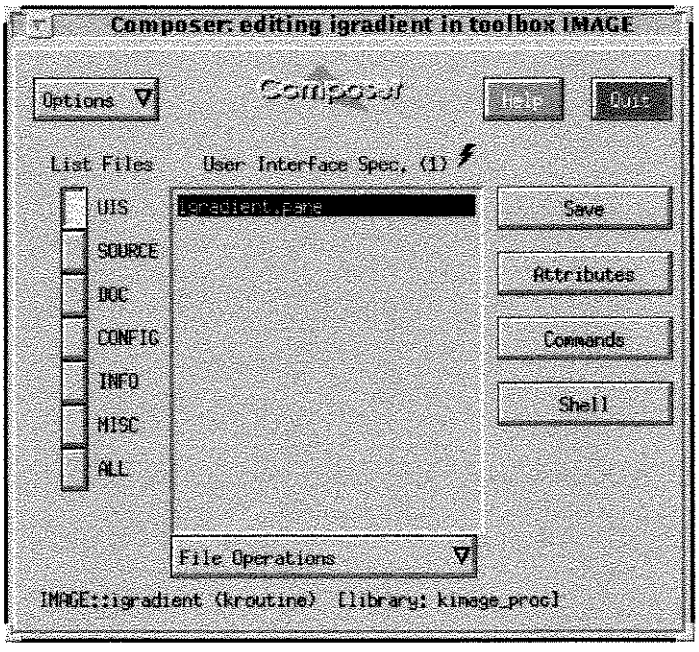

Figura 3.3: Composer : ferramenta para editar, manipular e compilar programas.

3.4. Composer

A ferramenta Composer é usada para editar, manipular e compilar programas (software objects) existentes. Os programas (software objects) são criados pelo Craftsman. Todos os programas (software objects) possuem um número de atributos associados a eles, que são usados para descrever o programa (object). O Composer tambem pode ser usado para modificar este conjunto de atributos.

A Figura 3.3 mostra a janela do Composer. Na parte inferior da janela, pode-se ver o nome da toolbox em letras maiúsculas seguido pelo nome do programa (object) e o tipo deste programa (object). O lado esquerdo mostra uma coluna de botões que controlam que tipo de arquivos irá aparecer na lista de arquivos que fica no centro da janela. O botão "File Operation" na parte inferior da lista de arquivos permite que alguns destes arquivos sejam vistos ou editados.

3.4.1. Tipos de arquivos

A coluna de botões no lado esquerdo da janela seleciona o tipo de arquivos que serão mostrados na lista de arquivos que fica no centro da janela.

UIS mostra somente arquivos UIS (User Interface Specification).

SOURCE mostra todos os códigos fontes.

DOC mostra as documentações sobre o programa (object). Como arquivos de ajuda e páginas do manual deste programa (object).

CONFIG lista o Makefiles do programa (object) e arquivo de configuração.

INFO mostra todos os arquivos de informação.

MISC mostra vários tipos de arquivos associados ao programa (object).

ALL mostra todos os arquivos que estão contidos no programa (object).

3.4.2. Atributos do programa

O botão "Attributes" traz uma janela quando selecionado que é usada para modificar atributos do programa (object).

3.4.3. Operações de arquivos

O botão "File Operations", quando selecionado, apresenta um menu de algumas operações de arquivos, tais como editar, adicionar, apagar, copiar e outras.

3.5. Guise

Guise, Figura 3.4, é uma ferramenta de interface gráfica para o usuário usada por todos os programas do Khoros. O Guise permite criar ou modificar um arquivo UIS (User Interface Specifcation). O arquivo UIS define a interface gráfica para o usuário (GUI-Graphical User Interface) e/ou a interface por linha de comando para o usuário

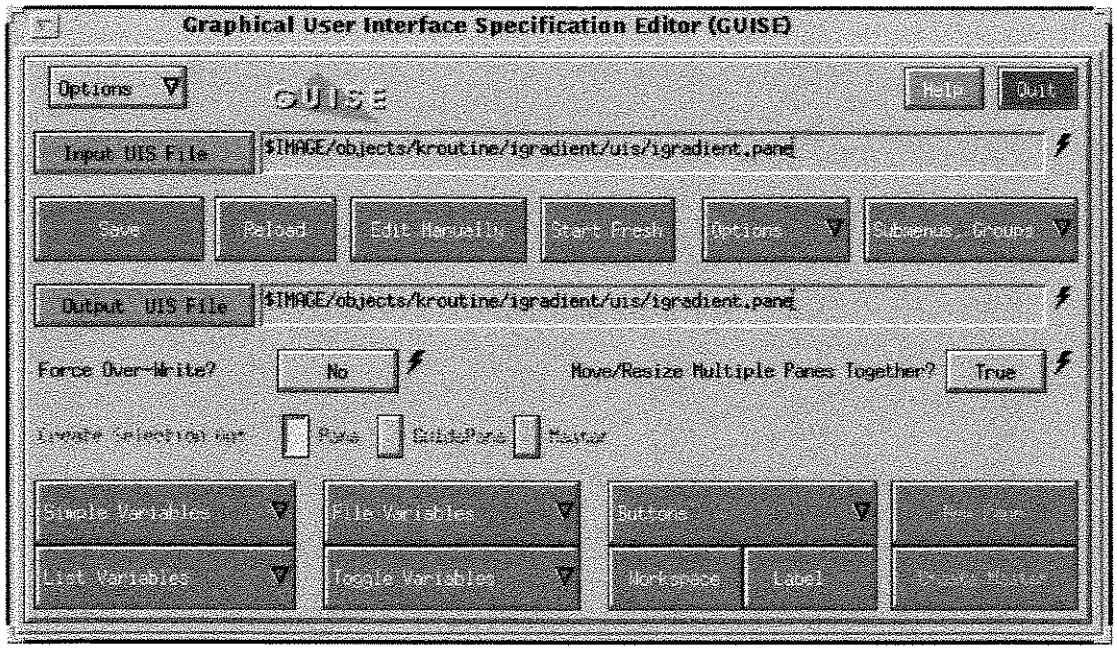

Figura 3.4: Guise : ferramenta para a manipulação da interface gráfica para o usuário.

(CLUI-Command Line User Interface. O Guise pode ser executado independentemente ou via Composer.

O Guise apresenta as seguintes características[1] para a criação e edição de GUI :

- Criação de todos os tipos de interfaces gráficas para o usuário
  1. input files, output files, stdin e stdout
  2. integer, float, double, string, flag e logical
  3. toggles de uma variedade de tipos de dados, como integer, float, double, string, input file e output file
  4. botões de action, routine, help e quit
  5. workspaces
  6. labels
- agrupamento de itens
- criação de submenus
- criação e manipulação da GUI com múltiplos panes e janelas
- e outras

3.6. Conclusão

Em síntese, o capítulo apresenta uma introdução ao ambiente de desenvolvimento de programas do sistema Khoros. Mostra algumas características, funções e elementos das ferramentas Craftsman, Composer e Guise para o desenvolvimento de programas[4][1].

Capítulo 4

Inserção de programas no ambiente Khoros

4.1. Introdução

O objetivo deste capítulo é sugerir procedimentos para a inserção de programas escritos em linguagem C no ambiente Khoros. Agora que já se tem uma visão geral do sistema Khoros e conhecimento da terminologia adotada, pode-se ilustrar como o ambiente de desenvolvimento de software pode ser usado para desenvolver um novo programa. Inicialmente, explica-se o propósito de cada arquivo associado com cada tipo de programa (software object), assim como várias questões relacionadas ao desenvolvimento de código usando o Khoros. A seguir, mostra-se um exemplo simples para alguns dos tipos de software object. Finalmente, apresenta-se a conclusão deste capítulo.

4.2. Arquivos associados com o programa

Um software object é simplesmente um conjunto de arquivos organizados numa estrutura de diretórios padronizada. Existem diferentes tipos de software objects : library objects, kroutine program objects, xvroutine program objects, pane objects e script objects. Os diretórios que podem estar contidos no software objects são : app-defaults, db, help, info, man, src e uis. Dependendo do tipo de software object, alguns dos diretórios

mencionados anteriormente podem não existir. Por exemplo, o tipo library object não possui o diretório uis. O diretório app-defaults existe somente no tipo xvroutine program object. Já o diretório db existirá em todos os tipos. Como na toolbox objects, alguns diretórios não serão criados até que sejam necessários. A Figura 4.1 mostra a estrutura associada com qualquer um dos tipos de software objects.

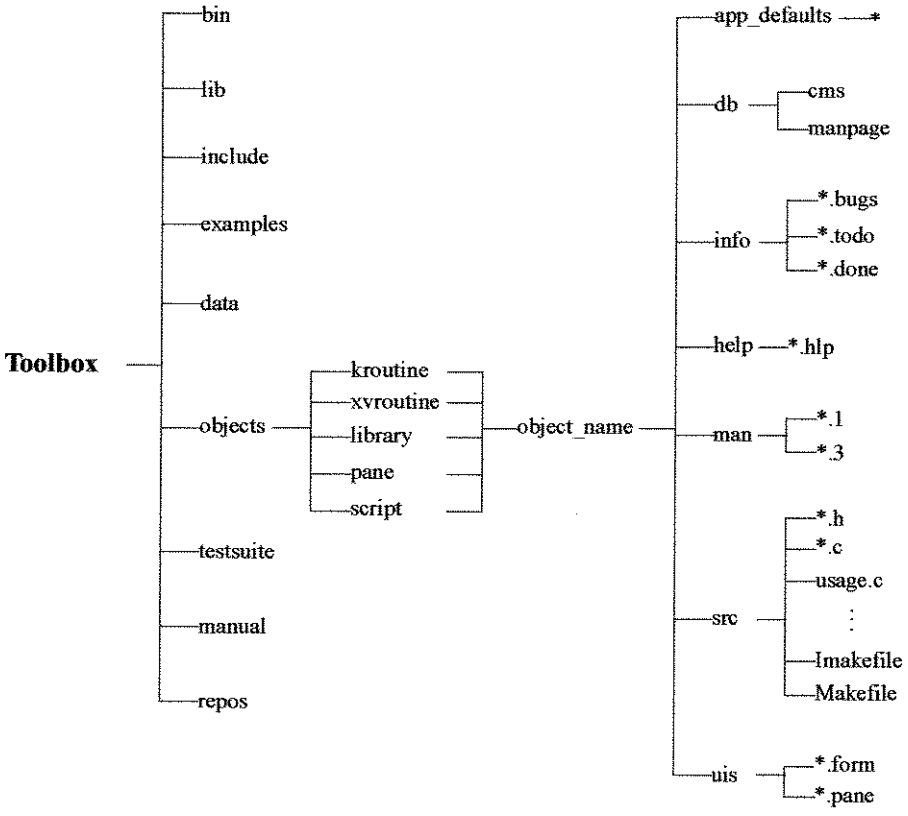

Figura 4.1: Estrutura de diretórios de uma toolbox incluindo a estrutura de subdiretórios associada aos software objects dessa toolbox.

- O **diretório app-defaults** é usado pelos tipos xvroutine program objects e library objects que utiliza interface gráfica de usuário ou programas visuais. Deve conter um arquivo de default que pode ser usado para fixar cores e outros atributos.
- O **diretório db** contém o arquivo cms, que mantem informações sobre o tipo de software object e seus atributos. O arquivo *.cksum armazena informações adicionais sobre o tipo de software object. Estes arquivos nunca devem ser modificados

manualmente porque são usados pelas ferramentas de desenvolvimento de programas do Khoros para gerência de programas, sincronismo das ferramentas e outras funções.

- O **diretório help** deverá conter o arquivo *.hlp para os tipos program objects. Este arquivo fornecerá ajuda quando acessado pelo Cantata.
- No **diretório info** existem arquivos de informação associados aos software objects, arquivos que documentam bugs conhecidos (*.bugs), listas de adição (*.todo) e desenvolvimentos recentes (*.done).
- No **diretório man** encontram-se as páginas man. Dependendo se o software object é um program object ou library object, estes arquivos deverão ser chamados *.1 ou *.3 respectivamente.
- O **diretório src** contém o código fonte para o program object ou library object, assim como o Imakefile e Makefile.
- O **diretório uis** contém um ou mais arquivos de especificação de interface de usuários (UIS-User Interface Specification) que estão associados com o software object. Para kroutine e pane objects, existirá somente o arquivo *.pane; para xvroutine também existirá um arquivo *.form e possivelmente um ou mais arquivos *.subform. Library objects somente usarão este diretório se for implementada uma interface gráfica de usuário (GUI-Graphical User Interface) ou programas visuais. Se o Script object for integrado no Cantata, existirá um arquivo *.pane neste diretório

Existem questões que são comuns na maioria dos tipos de software object. São elas :

1. A criação da interface gráfica de usuário (GUI - Graphical User Interface) definida pelo arquivo de especificação de interface de usuário (UIS-User Interface Specification);
2. A adição de código e documentação aos arquivos gerados automaticamente;
3. Os arquivos database que são usados para gerência dos software objects;
4. Os Imakefiles e Makefiles que são usados para compilar o programa e
5. Os arquivos de documentação.

4.2.1. A interface gráfica de usuário

A interface gráfica de usuário (GUI-Graphical User Interface) está organizada numa hierarquia de 3 níveis :

1. Um master form (opcional),
2. Um ou mais subforms individuais e
3. Um ou mais panes de cada subform

O uso destas interfaces hierárquicas ajuda os usuários organizar os problemas dentro de classes e subclasses.

Todos os tipos de software objects possuem pelo menos um arquivo de especificação de interface de usuários (UIS-User Interface Specification). O arquivo UIS consiste de uma série de linhas estritamente definidas e cada tipo de linha contém a informação necessária para descrever completamente o tipo de item na interface gráfica de usuário (GUI-Graphical User Interface), assim como na interface de usuário de linha de comando (CLUI- Command Line User Interface). O arquivo UIS é usado para três propósitos diferentes no Khoros :

1. Definir a interface de usuário de linha de comando (CLUI- Command Line User Interface) de todos os programas do Khoros.
2. Definir a interface gráfica de usuário (GUI-Graphical User Interface) do programa .
3. Definir uma interface gráfica de usuário (GUI-Graphical User Interface) interativa para xvroutines.

4.2.2. Documentação e códigos gerados automaticamente

O sistema de desenvolvimento de programas do ambiente Khoros usa dois geradores de códigos para a geração automática de código e documentação para os softwares objects. O gerador de códigos Ghostwriter gera código e documentação para todos os software objects do Khoros, enquanto o gerador de códigos Conductor é somente usado com xvroutines. Tanto o Ghostwriter como o Conductor são chamados através do botão "Generate Code" do menu Commands da janela do Composer.

Quando o Ghostwriter é usado com um program object, ele gera o main, o código para obter os argumentos da linha de comando, a rotina usage e o esqueleto da documentação para o programa. Quando usado com library object, ele gera as páginas man de cada rotina de biblioteca pública assim como a página man para a biblioteca como um todo.

O Conductor é uma ferramenta de geração de códigos usada somente para xvroutine.

Muitos arquivos gerados pelo Ghostwriter precisam ser modificados. Estes arquivos são gerados novamente todas as vêzes que o Ghostwriter for executado. Para isso existem pares de tags no código destes arquivos para preservar a adição. Usando corretamente os tags, pode-se adicionar código e documentação no programa sem a preocupação de perder as mudanças feitas quando o Ghostwriter é executado. Existem duas regras [4] para usar os arquivos gerados pelo Ghostwriter :

1. Nunca apagar ou modificar os tags gerados pelo Ghostwriter e
2. Sempre adicionar texto ou código entre os pares de tags.

Dependendo da localização dos tags, eles podem ser óbvios ou camuflados. Os tags gerados no arquivo fonte são óbvios enquanto os gerados na documentação são camuflados.

4.3. Criar uma Toolbox

O software object deve ser criado dentro de uma toolbox. Esta toolbox pode ser uma já existente ou uma criada pelo usuário. Costuma-se criar uma toolbox para os software objects do usuário. Esta seção mostra os passos para a criação de uma toolbox.

1. Executar o Craftsman da linha de comando :

   % craftsman

2. Selecionar a opção "Create Toolbox" do menu "Toolbox Operation".
3. Preencher os campos do subform "Create a new toolbox"
4. Pressionar o botão "Create Toolbox"

Depois de criada uma toolbox, pode-se alterar seus atributos selecionando-se a opção "Toolbox Attributes" do menu "Toolbox Operations".

A seguir, apresenta-se um exemplo de como criar uma toolbox :

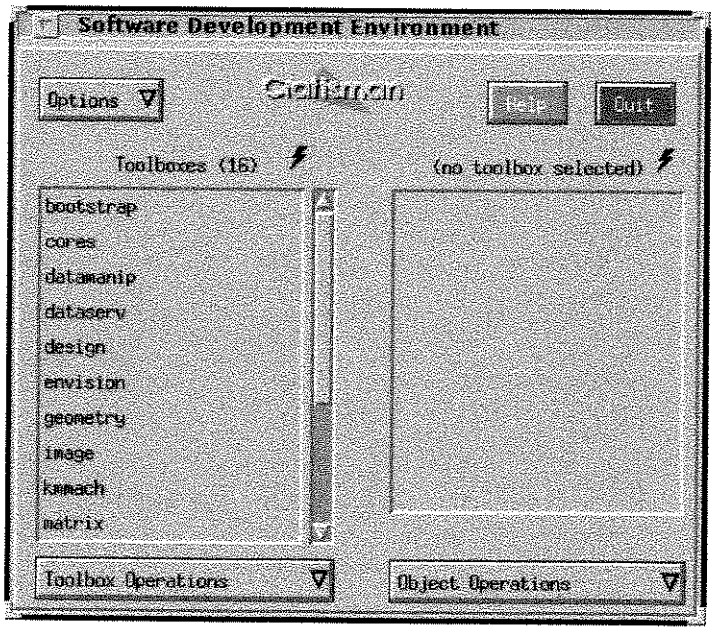

Figura 4.2: Craftsman : Ferramenta de gerência de programa e toolbox.

Deve-se primeiro criar uma toolbox para conter o novo programa. Para criar uma toolbox, usa-se a ferramenta Craftsman. Executando a ferramenta Craftsman aparece a janela ilustrada na da Figura 4.2.

A janela do Craftsman apresenta duas listas, sendo que a da esquerda corresponde a uma lista de toolboxes e a da direita a uma lista de programas (software objects). Os programas (software objects) de uma determinada toolbox aparecem na lista da direita quando esta toolbox é selecionada. Abaixo da lista de toolboxes e da lista de programas (software objects), existem os botões "Toolbox Operations" e "Object Operations", respectivamente.

Seleciona-se, do menu "Toolbox Operation", a opção "Create Toolbox". Uma janela idêntica à Figura 4.3 irá aparecer.

Sobre os campos da janela "Create a New Toolbox" :

1. Toolbox Name : Entrar com um nome para a toolbox.
2. Toolbox Path : Entrar com um path para a toolbox.
3. Toolbox Title : Entrar com um título para a toolbox.
4. Toolbox Author : Obtido do arquivo .khoros_env
5. Toolbox Status : As componentes das opções Scratch e Work não são inseridas no

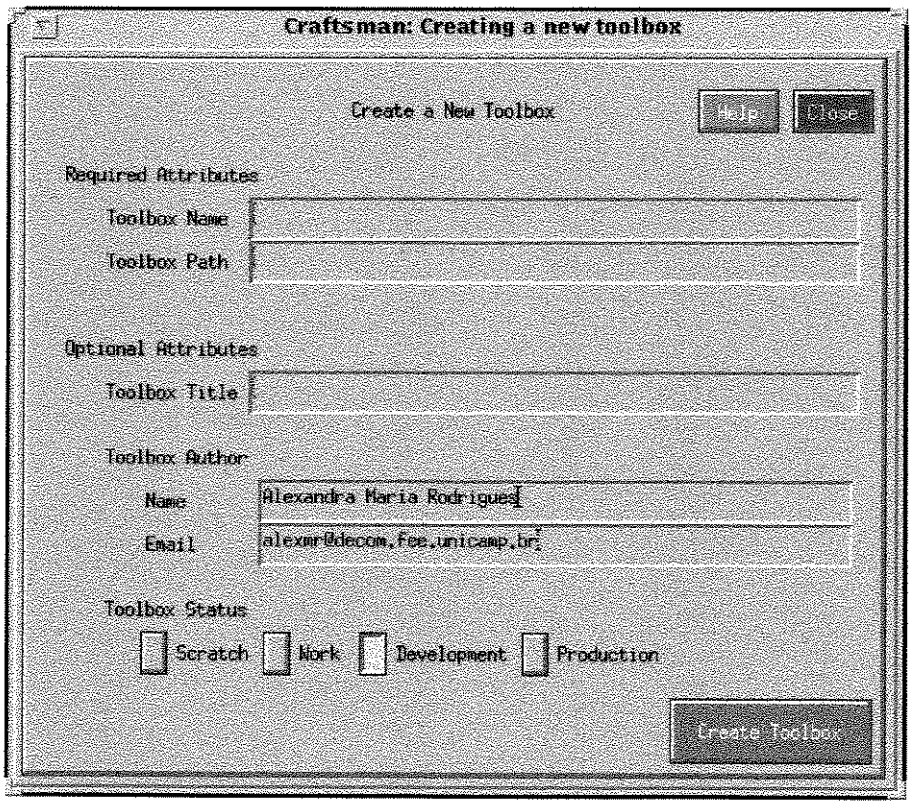

Figura 4.3: Janela para criação de uma toolbox.

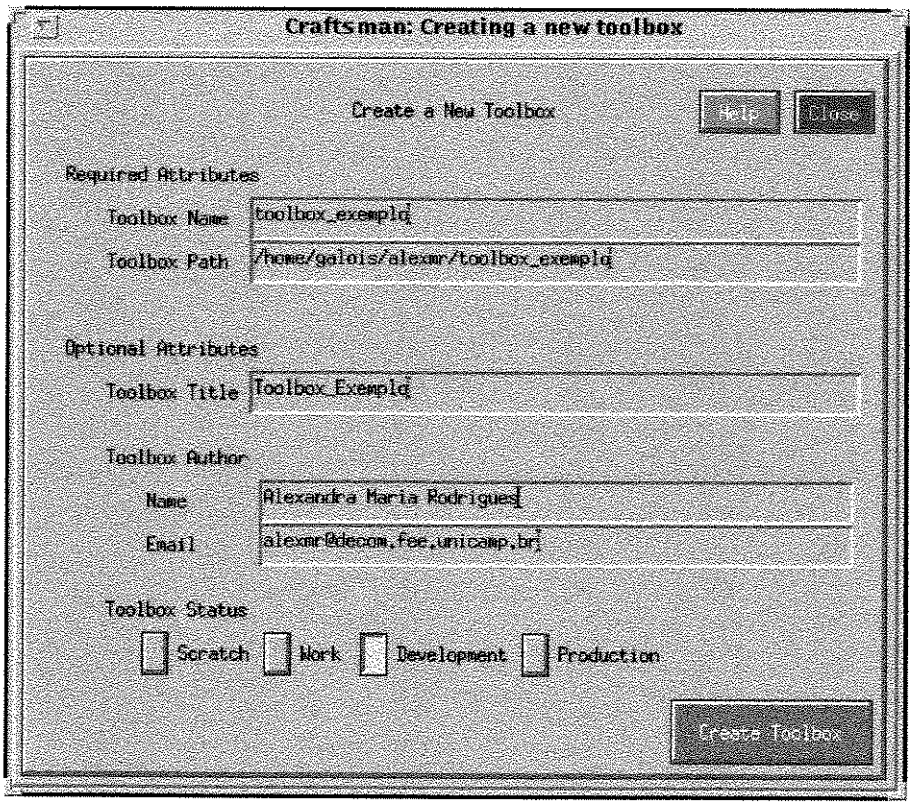

Figura 4.4: Janela para criação de uma toolbox com os campos preenchidos.

Cantata.

Cria-se uma nova toolbox chamada Toolbox_Exemplo, preenchendo os campos da janela "Create a New Toolbox" e pressionando o botão "Create Toolbox" no canto inferior direito da janela. Antes de pressionar o botão "Create Operation", a janela deverá ser parecida com a da Figura 4.4.

Os atributos de uma toolbox podem ser modificados através da opção "Toolbox Attributes" do menu "Toolbox Operations". As janelas "Toolbox Operations" são mostradas nas Figuras 4.5, 4.6, 4.7, 4.8 e 4.9.

4.4. Library Objects

Existem questões que são particulares da Library objects. São elas :

1. Passos para o desenvolvimento da library object

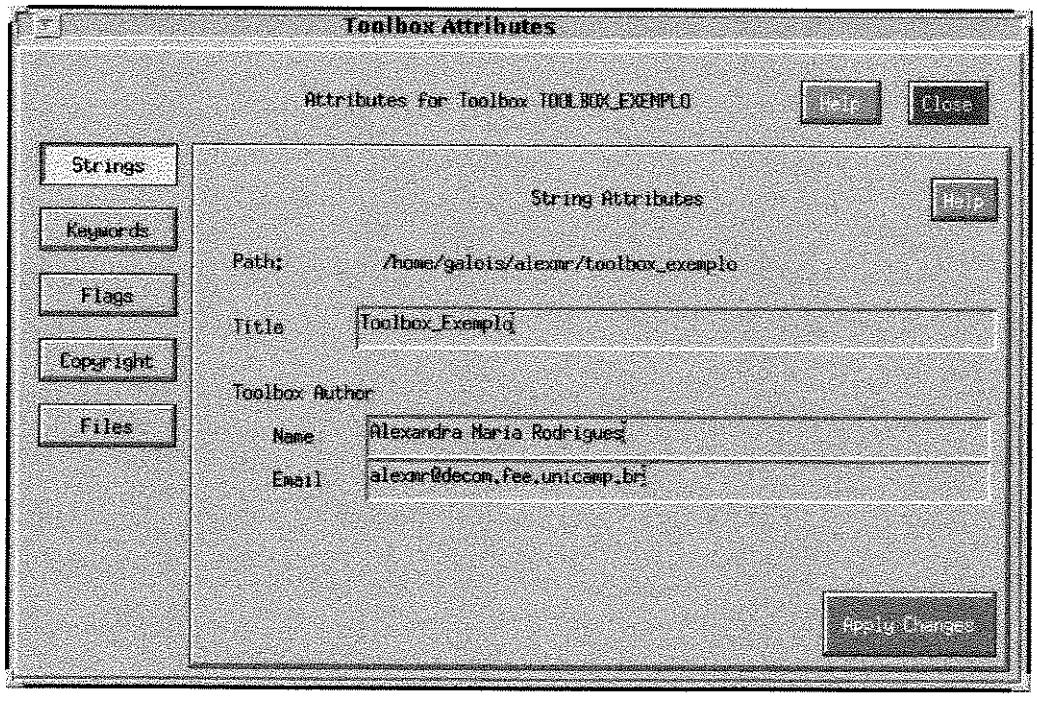

Figura 4.5: Janela de atributos da toolbox com a opção "Strings" selecionada.

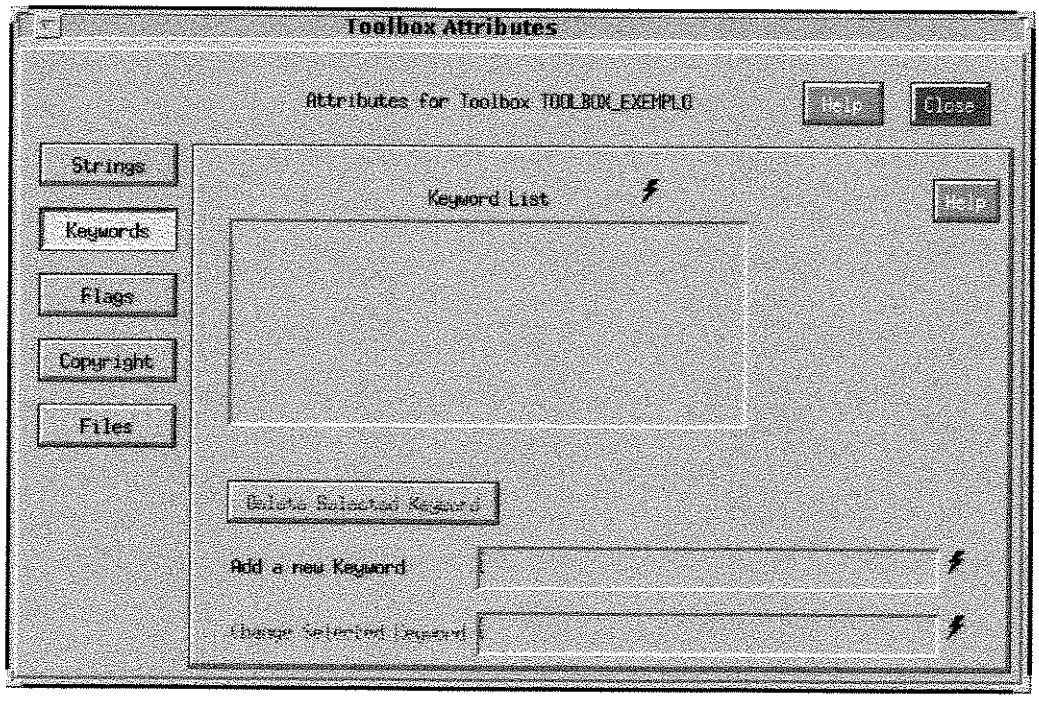

Figura 4.6: Janela de atributos da toolbox com a opção "Keywords" selecionada.

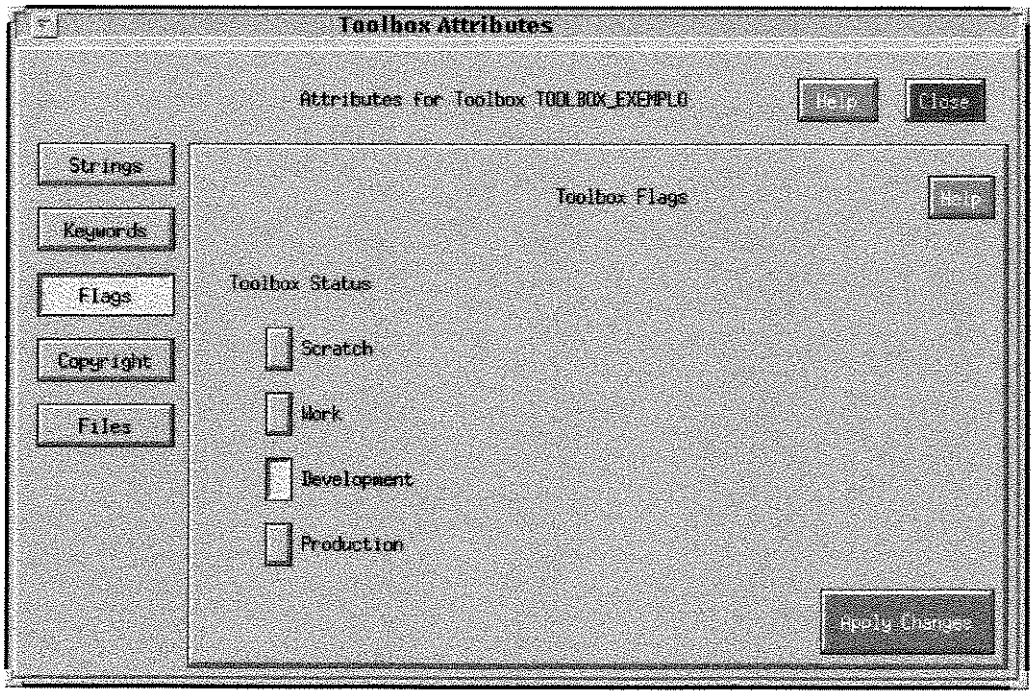

Figura 4.7: Janela de atributos da toolbox com a opção "Flags" selecionada.

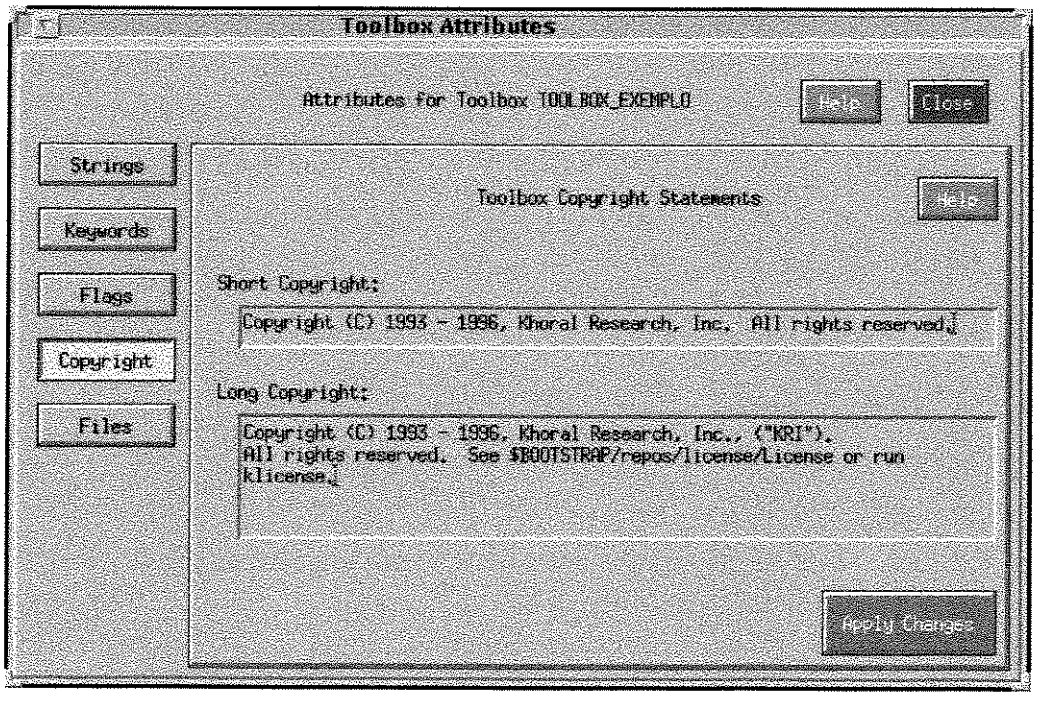

Figura 4.8: Janela de atributos da toolbox com a opção "Copyright" selecionada.

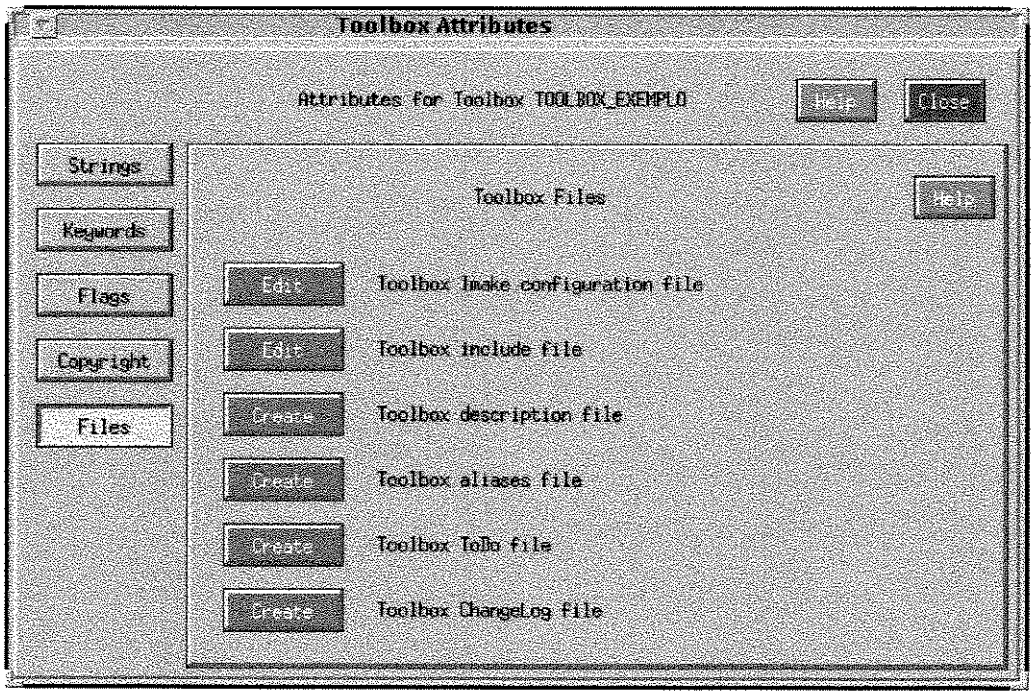

Figura 4.9: Janela de atributos da toolbox com a opção "Files" selecionada.

2. As diferenças entre rotinas separadas e lkroutines
3. As diferenças entre rotinas pública, privada e estática
4. Itens do código fonte do arquivo de biblioteca
5. Páginas man para as rotinas de biblioteca

4.4.1. Passos para o desenvolvimento da library object

1. Selecionar ou criar uma toolbox para a biblioteca,
2. Criação da Library Object e
3. Edição da biblioteca.

Selecionar a toolbox para a biblioteca

Usa-se o Craftsman para se criar uma nova toolbox, como explicado na seção anterior, ou pode-se escolher uma toolbox já existente. Na toolbox criada ou escolhida deve-se criar a biblioteca.

Executa-se o Craftsman da linha de comando. Seleciona-se uma toolbox já existente para criar sua biblioteca, selecionando com o mouse uma opção da lista de toolbox no lado esquerdo da janela do Craftsman.

As rotinas de biblioteca podem ser divididas em duas categorias : as lkroutines e as rotinas soltas, e em três tipos : Pública, Privada e Estática. Recomenda-se o manual [4] para maiores detalhes.

Criação da Library Object

Depois de selecionada a toolbox desejada para o pane object da lista de "Toolboxes", selecionar a opção "Create Object" do menu "Object Operations" para que apareça o subform "Create new object".

Preencher os campos do subform "Create new object" e pressionar o botão "Create LIBRARY".

Edição da biblioteca

Através do Composer, pode-se adicionar novos arquivos na biblioteca. Para isto, basta selecionar a opção "Add new files" do menu "File Operations" e mencionar o arquivo a ser adicionado na biblioteca.

Para editar o arquivo *.c (rotina pública para a biblioteca), seleciona-se o arquivo *.c da lista de arquivos da opção "SRC" e seleciona-se a opção "Edit" do menu "File Operations" para chamar o editor de texto. O mesmo pode ser feito para editar o arquivo *.h.

Usando a opção "Make Install" do menu "Make..." do subform "Commands" do Composer, pode-se compilar a rotina de biblioteca.

4.4.2. As diferenças entre rotinas separadas e lkroutines

Algumas rotinas de biblioteca são geradas em conjunto com uma particular kroutine, para conter funções para a kroutine de maneira que estas funções possam ser usadas por outros programas do sistema. As rotinas de bibliotecas nomeadas depois da kroutine usam

a letra l antes do nome da kroutine, por essa razão são chamadas de lkroutines.

Outras rotinas de biblioteca não estão associadas com uma particular kroutine. São criadas para conter funções necessárias para outras partes do sistema. As rotinas de biblioteca separadas ou soltas são nomeadas pelo usuário, devendo ter prefixos que as identifiquem com relação a qual biblioteca elas pertecem.

A diferença entre essas duas rotinas de biblioteca é que, para a rotina lkroutine, existem tags que cercam todo o seu corpo. Já para uma rotina solta não existem tags envolvendo sua rotina. Outra diferença é que a rotina solta de biblioteca pertence somente à biblioteca em que ela aparece e a lkroutine é considerada parte de dois software objects : a kroutine com o qual a ela está associada e a biblioteca a qual ela pertence.

4.4.3. As diferenças entre rotinas pública, privada e estática

As lkroutines são, por definição, rotinas públicas. Por outro lado, as rotinas separadas ou soltas podem chamar funções privadas ou estáticas dentro de uma mesma biblioteca.

A rotina pública é uma função criada para ser usada por software objects de outras bibliotecas. A rotina privada é uma função que é criada para não ser usada fora da biblioteca a qual pertence. Seu cabeçalho é diferente da rotina pública e não pode ser usado para gerar a página man. A rotina estática é parecida com a rotina privada. A diferença é que a rotina estática é criada para não ser usada fora do arquivo ao qual pertence.

4.4.4. Itens do código fonte do arquivo de biblioteca

Os itens a seguir aparecem na parte superior do arquivo de biblioteca. Estes itens são automaticamente gerados quando uma biblioteca é criada.

Cabeçalho RCS : O código fonte dos arquivos de biblioteca começa com o cabeçalho RCS, que é mostrado na Figura 4.10.

Copyright : Se existir um copyright, ele será colocado depois do cabeçalho RSC. Através do Craftsman, pode-se criar um copyright do usuário.

O cabeçalho do arquivo de biblioteca : Todo código fonte do arquivo de biblioteca deve

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif
```

Figura 4.10: Cabeçalho RSC para os arquivos de uma biblioteca.

ter um cabeçalho depois do copyright como o da Figura 4.11.

```
/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>> Library Routine for #object-name#
  >>>>
  >>>> Private:
  >>>> Static:
  >>>> Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */
```

Figura 4.11: Cabeçalho do arquivo de biblioteca

A declaração #include internals.h : Toda biblioteca possui um arquivo internals.h, Neste arquivo as rotinas são declaradas, as estruturas são definidas, os macros são escritos e assim por diante. Depois do cabeçalho da biblioteca, todo código fonte da biblioteca deve ter a declaração como mostrada na Figura 4.12 :

```
#include "intervals.h"
```

Figura 4.12: Declaração do arquivo internals.h.

Cabeçalho da rotina pública de biblioteca : O cabeçalho de rotina pública é padronizado porque a página man será gerada automaticamente das informações fornecidas por ele. O cabeçalho de rotina pública é mostrado na Figura 4.13.

Cabeçalho da rotina privada/estática de biblioteca : Seu cabeçalho é mostrado na Figura 4.14.

```
/*****************************************************************
*
* Routine Name: function_name - short description of function
*
*     Purpose: This should be a complete description that anyone
*              could understand;  it should have acceptable grammar
*              and correct spelling.
*
*                    This is the header for a PUBLIC function
*
*      Input: argument1 - explanation
*             argument2 - explanation
*             argument3 - explanation
*
*     Output: argument4 - explanation
*             argument5 - explanation
*
*    Returns: TRUE (1) on success, FALSE (0) otherwise
*
* Restrictions: Restrictions on data or input as applicable
*  Written By:
*       Date: Nov 15, 1994
*    Verified:
* Side Effects:
*Modifications:
*****************************************************************/
```

Figura 4.13: Cabeçalho de uma rotina pública.

```
/*-------------------------------------------------------------

 Routine Name: foo - short description of foo()

    Purpose: This should be a complete description that anyone
             could understand;  it should have acceptable grammar
             and correct spelling.

                     This is the header for a PRIVATE or STATIC function

     Input: argument1 - explanation
            argument2 - explanation
            argument3 - explanation

    Output: argument4 - explanation
            argument5 - explanation

   Returns: TRUE (1) on success, FALSE (0) otherwise

  Written By:
       Date:
Modifications:

-------------------------------------------------------------*/
```

Figura 4.14: Cabeçalho de uma rotina privada ou estática de biblioteca.

4.4.5. Páginas man para as rotinas de biblioteca

As páginas man são encontradas no diretório man/ da biblioteca (library object). As bibliotecas possuem dois tipos de páginas man : uma página man para cada rotina pública de biblioteca e uma página man para a biblioteca como um todo.

Páginas man para rotinas públicas

A Figura 4.15 mostra uma página man para a rotina pública de biblioteca.

Páginas man para a biblioteca como um todo

A Figura 4.16 mostra a página man da biblioteca.

```
.TH "routine_name" "TOOLBOX" "LIBRARY" "(library)" "date"
.nr LL 7.0i
.SH LIBRARY ROUTINE
routine_name - short description
.SH LIBRARY CALL
.nf
int routine_name(
  data_type arg1,
  data_type arg2,
  data_type arg3,)
.fi
.SH INPUT
.IP "arg1 -" 18
description of input argument
.IP "arg2 -" 18
description of input argument
.IP "arg3 -" 18
description of output argument
.SH RETURN VALUE
Return value description
.SH DESCRIPTION
Long description of library routine
.SH ADDITIONAL INFORMATION
.SH EXAMPLES
.SH SIDE EFFECTS
.SH RESTRICTIONS
.SH MODIFICATION
.SH FILES
$TOOLBOX/objects/library/library_name/src/filename.c
.SH SEE ALSO
library(3)
.SH COPYRIGHT
Copyright (C) 1993 - 1996, Khoral Research, Inc.  All rights reserved.
```

Figura 4.15: Página man para rotinas públicas de biblioteca.

```
.TH "library_name" "TOOLBOX_NAME" "COMMANDS" "" "date"
.SH NAME
binary_name \- short library description
.SH DESCRIPTION
.SH "LIST OF LIBRARY FUNCTIONS"
.whatis library_name public_routine_one
.whatis library_name public_routine_two
.whatis library_name public_routine_three
.SH "ADDITIONAL INFORMATION"
.SH "LOCATION OF SOURCE FILES:"
$TOOLBOX/objects/library/library_name/src
.SH "LOCATION OF PUBLIC INCLUDE FILE:"
$TOOLBOX/include/library_name/library_name.h
.SH "YOU MUST INCLUDE:"
#include <toolbox.h>
.SH "SEE ALSO"
.SH "SEE MANUAL"
.SH COPYRIGHT
Copyright (C) 1993 - 1996, Khoral Research, Inc.  All rights reserved.
```

Figura 4.16: Página man para a biblioteca.

4.5. Program Objects

Todos os program objects, tanto uma xvroutine como uma kroutine, devem ter um arquivo UIS *.pane. Este arquivo descreve a interface de usuário de linha de comando do programa.Usando as informações deste arquivo, o ghostwriter gera vários arquivos para o programa object :

Arquivo *.c Gerado no diretório src/, este arquivo contem código gerado automaticamente pelo Ghostwriter e também pares de tags onde devem ser colocados o código do usuário.

Arquivo usage.c Também no diretório src/, este arquivo é gerado automaticamente pelo Ghostwriter. Este arquivo contem a rotina program_get_args(), que obtem os argumentos do progama do usuário via linha de comando.

Arquivo *.h Gerado no diretório src/ do program object, o ele é o principal arquivo de include para o programa. Neste arquivo encontra-se a estrutura de informação da interface de usuário por linha de comando.

Arquivo *.1 O Ghostwriter gera uma página man1 no diretório man/ do program object, que contém comandos formatados de acordo com as especificações do sistema

Khoros.

Arquivo l*.c Este arquivo somente existirá se uma kroutine for criada com uma biblioteca associada a ela.

Arquivo *.hlp Gerado no diretório help/, este arquivo é uma página de help que está disponível ao usuário quando acessada pelo Cantata.

4.5.1. Passos para o desenvolvimento de uma kroutine

1. Selecionar ou criar uma toolbox para a kroutine.
2. Na toolbox selecionada ou criada, através do Craftsman, criar a kroutine.
   (a) Selecionar a toolbox desejada para a kroutine da lista de "Toolboxes".
   (b) Selecionar a opção "Create Object" do menu "Object Operations" para que apareça o subform "Create new object".
   (c) Preencher os campos do subform "Create new object" e pressionar o botão "Create KROUTINE".
3. Editar a kroutine.
   (a) Especificar os argumentos da linha de comando
      i. Selecionar a kroutine criada da lista de software object da janela do Craftsman.
      ii. Selecionar a opção "Edit Object (Composer)" do menu "Object Operations" para que apareça a janela do Composer.
      iii. Da janela do Composer, pressionar o botão "UIS" e selecionar o arquivo *.pane.
      iv. Selecionar a opção "Guise" do menu "File Operations".
      v. Especificar de forma interativa os argumentos da linha de comando do pane no arquivo *.pane.
      vi. Depois das alterações no arquivo *.pane, gerar novamente o código usando o botão "Generate Code" do subform "Commands" da janela do Composer.

(b) Editar o código da kroutine

i. Da janela do Composer, pressionar o botão "SRC" e selecionar o arquivo *.c

ii. Abrir o editor de texto através da opção "Edit" do menu "File Operations" da janela do Composer e fazer as modificações necessárias para o funcionamento do código.

iii. Se existir uma lkroutine associada a kroutine, essa lkroutine também deverá ser editada. Para editar um arquivo l*.c deve-se seguir os passos i, selecionando o arquivo l*.c, e ii anteriores.

4. Através do Composer, compilar e instalar a kroutine

(a) Selecionar o subform "Commands". Pressionar o botão "Generate Code" para que o código da kroutine seja atualizado.

(b) Do subform "Commands" selecione a opção "Make Install" do menu "Make..."

5. Escrever uma página man. Editar esta página e através do Ghostwriter gerar automaticamente a página help.

6. Executar a kroutine da linha de comando e verificar os resultados.

7. Executar a kroutine dentro do Cantata.

4.5.2. Exemplo de uma kroutine

Selecionar a toolbox para a kroutine

Execute o Craftsman da linha de comando. Depois de criada a toolbox *toolbox_name* como mostrado na seção 4.3, selecione esta toolbox colocando o cursor do mouse sobre o nome desta toolbox e pressionando o botão esquerdo do mouse. A janela do Craftsman deverá parecer-se com a da Figura 4.17.

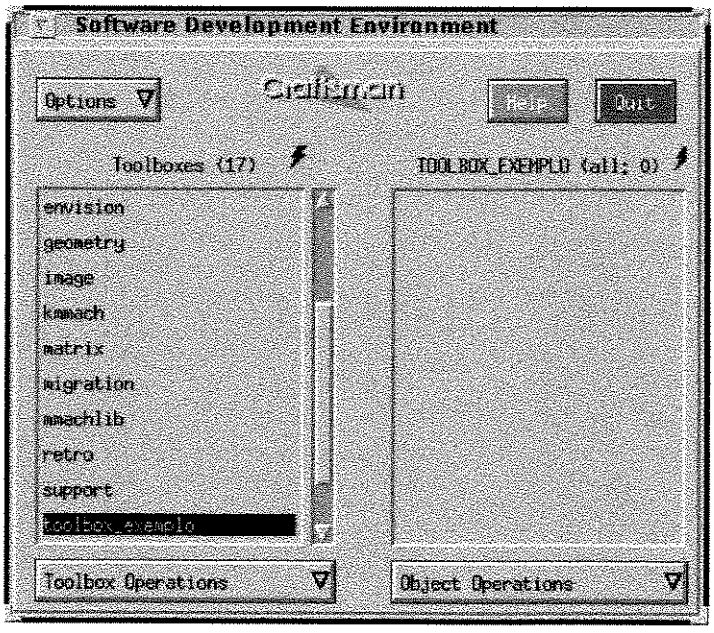

Figura 4.17: Janela do Craftsman com a toolbox *toolbox_exemplo* selecionada.

Através do Craftsman criar a kroutine

Para criar o novo programa chamado k_teste, deve-se selecionar a opção "Create Object" do menu "Object Operations" da janela do Craftsman. Uma vez selecionada esta opção aparecerá a janela "Creating new object in toolbox ..."como a da Figura 4.18.

Selecionado o tipo kroutine como mostra a figura anterior, os seguintes campos devem ser preenchidos.

- **Object Name :** nome para o programa
- **Binary Name :** nome para o programa binário
- **Icon Name :** nome do ícone no Cantata
- **Author :** fornecido pelo arquivo .khoros_env
- **Email Address :** fornecido pelo arquivo .khoros_env
- **Category :** nome que aparece no menu principal do botão "Glyphs" da janela do Cantata.
- **Subcategory :** nome que aparece no submenu do botão "Glyphs" da janela do Cantata.

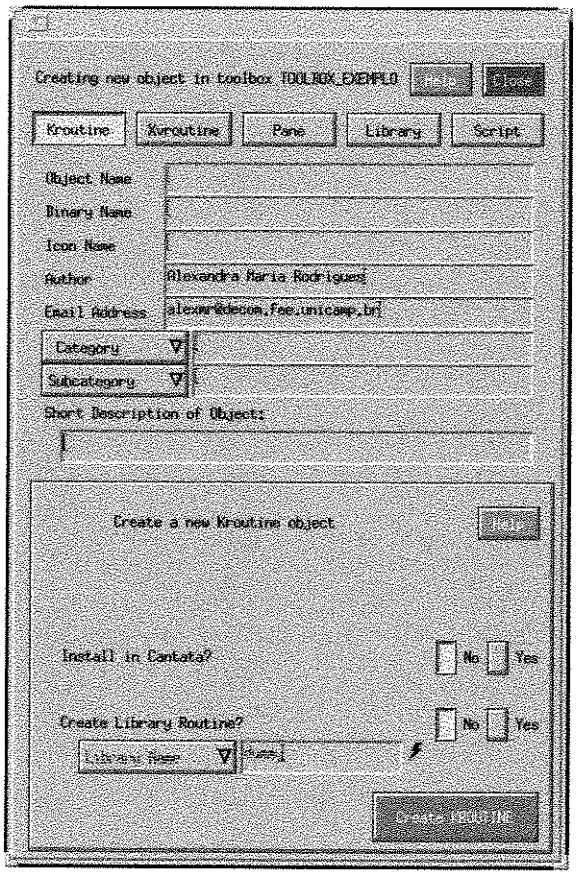

Figura 4.18: Janela para a criação de um novo programa.

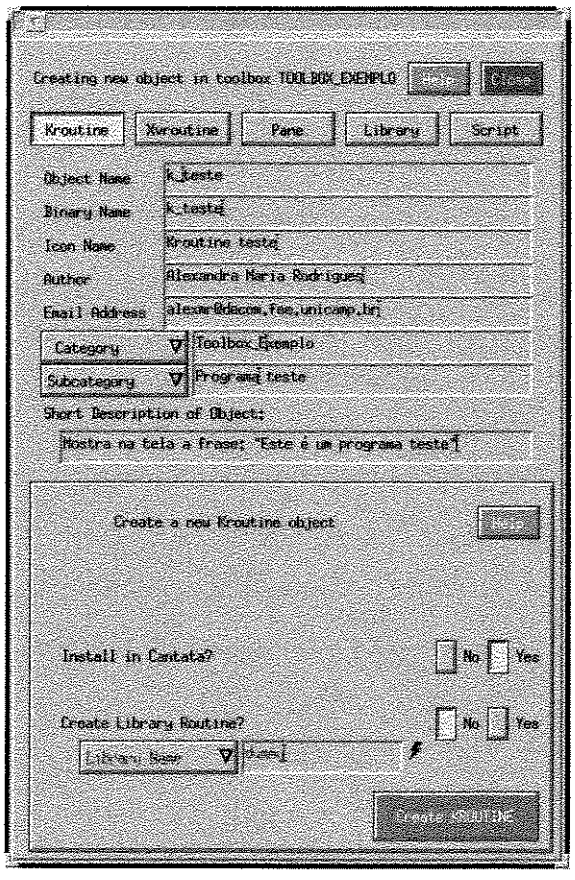

Figura 4.19: Janela para a criação de um novo programa com os campos preenchidos.

- **Short Description of Object** : pequena descrição do programa
- **Install in Cantata ?** : Seleciona-se a opção "Yes" para que o programa seja instalado no Cantata.
- **Create Library Routine ?** : Seleciona-se a opção "No" para que não seja criado um arquivo de biblioteca associado ao programa.

A janela com os campos preenchidos deverá parecer-se com a da Figura 4.19.

Depois de preenchidos todos os campos da janela, pressiona-se o botão "Create kroutine" para que o novo programa seja criado.

Editar a kroutine

O Craftsman gera o arquivo *.pane e vários outros arquivos baseados no tipo de programa (program object) quando um programa é criado. Primeiro, precisa-se editar o

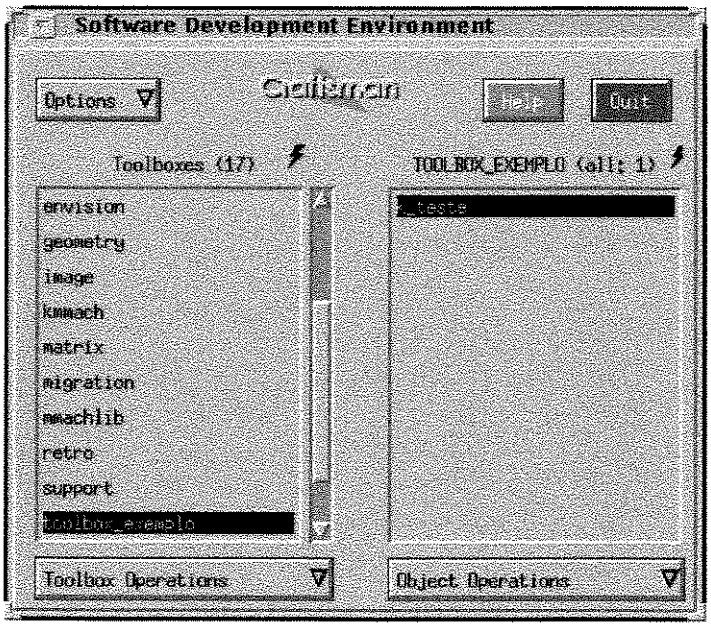

Figura 4.20: Janela do Craftsman com o program object k_teste, no caso uma kroutine, selecionado.

arquivo de interface para o usuário, o arquivo *.pane. Para isso, com o programa selecionado, como mostra a Figura 4.20, pressiona-se o botão "Object Operation" e escolhe-se a opção "Edit Object (Composer)".

A janela do Composer aparecerá como mostra a Figura 4.21.

Seleciona-se o botão "File Operations" e escolhe-se a opção "Guise" para a edição do arquivo k_teste.pane. Duas janelas aparecerão como as da Figura 4.22 e a da Figura 4.23.

Para o programa k_teste, apenas será necessária a inserção de uma variável simples, uma string. Então deve-se remover as variáveis de arquivo de entrada e saída. Para remover a variável arquivo de entrada basta pressionar o botão do meio do mouse para que uma nova janela apareça na tela. Nessa janela pressione o botão "Delete" que removerá esta variável arquivo de entrada. Do mesmo modo, também remove-se a variável arquivo de saída. Depois de removidas as variáveis de arquivo, o pane do programa parecerá com a Figura 4.24.

Para adicionar a variável string, deve-se pressionar o botão "Simple Variables" da janela do Guise e selecionar a opção "String". Agora a janela do arquivo new_program.pane deverá parecer-se com a Figura 4.25.

Para mudar os atributos da variável string basta pressionar o botão do meio do mouse

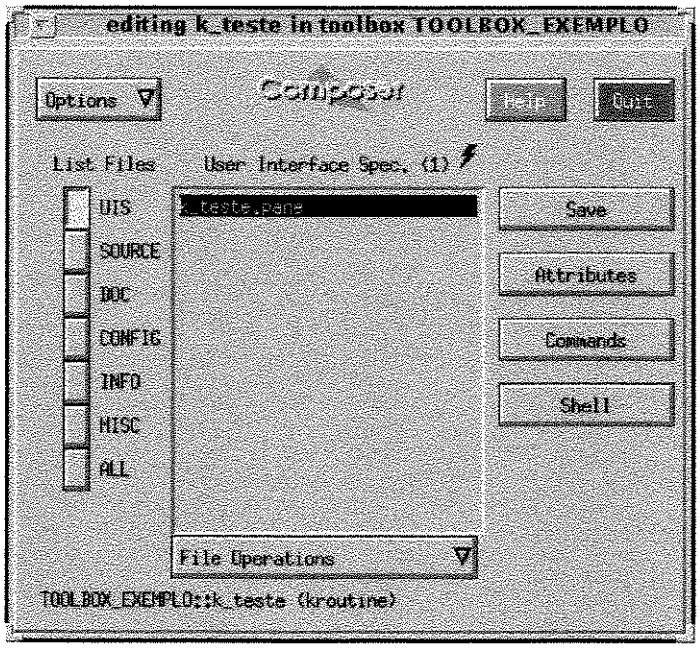

Figura 4.21: Composer : ferramenta de edição, manipulação e composição de programas.

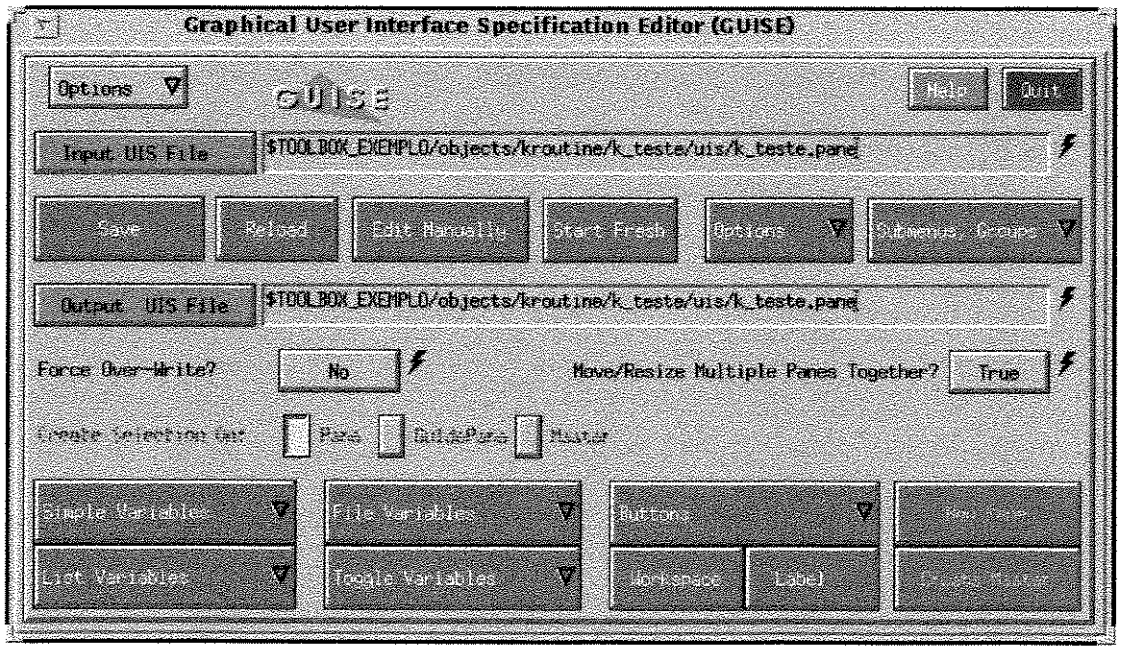

Figura 4.22: Guise : ferramenta de interface gráfica para o usuário.

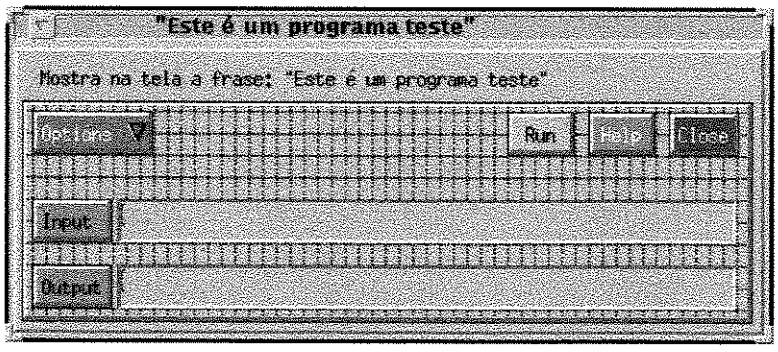

Figura 4.23: Interface gráfica de usuário.

Figura 4.24: Interface gráfica de usuário, o pane, com as variáveis de arquivo removidas.

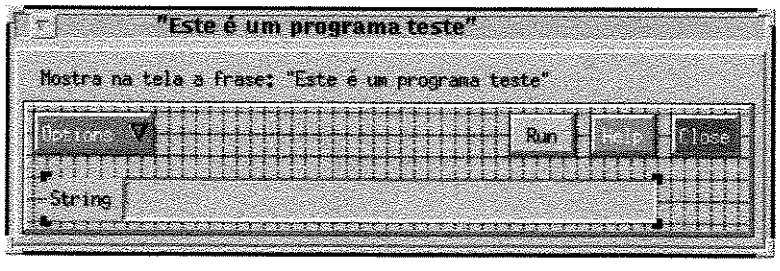

Figura 4.25: O pane da krroutine k_teste com já com a variável string adicionada.

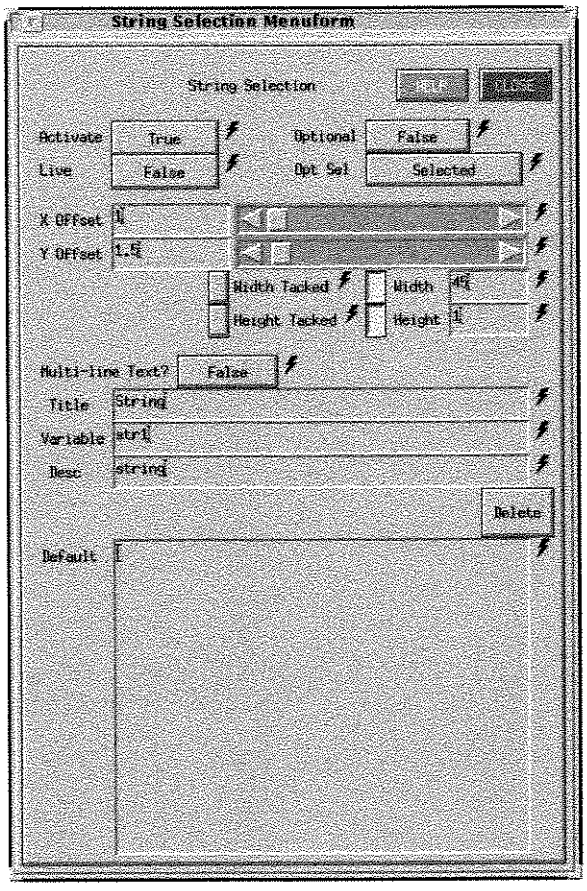

Figura 4.26: Janela da variável string, o lugar para se mudar os atributos dessa variável.

para aparecer a janela como a da Figura 4.26.

Deve-se preencher os campos desta janela como mostrado na Figura 4.27.

Então a interface gráfica de usuário deverá parecer-se com a Figura 4.28.

Para salvar todas as alterações feitas na interface gráfica de usuário, deve-se pressionar o botão "SAVE (Needed)" da janela do Guise como pode ser visto na Figura 4.29.

Depois das alterações no arquivo k_teste.pane, gera-se novamente o código usando o botão "Generate Code" do subform "Commands" da janela do Composer. A Figura 4.30 mostra o subform "Commands" depois de pressionado o botão "Generate Code".

Feitas as alterações da interface gráfica de usuário, o arquivo fonte será editado. O acesso ao arquivo fonte é feito pressionando-se o botão "SRC" no lado esquerdo da janela do Composer. No centro da janela, aparecerão todos os arquivos fontes associados ao programa (program object) "k_teste", como mostrado na Figura 4.31.

Para edtitar o arquivo k_teste.c basta selecioná-lo e pressionar o botão "File Opera-

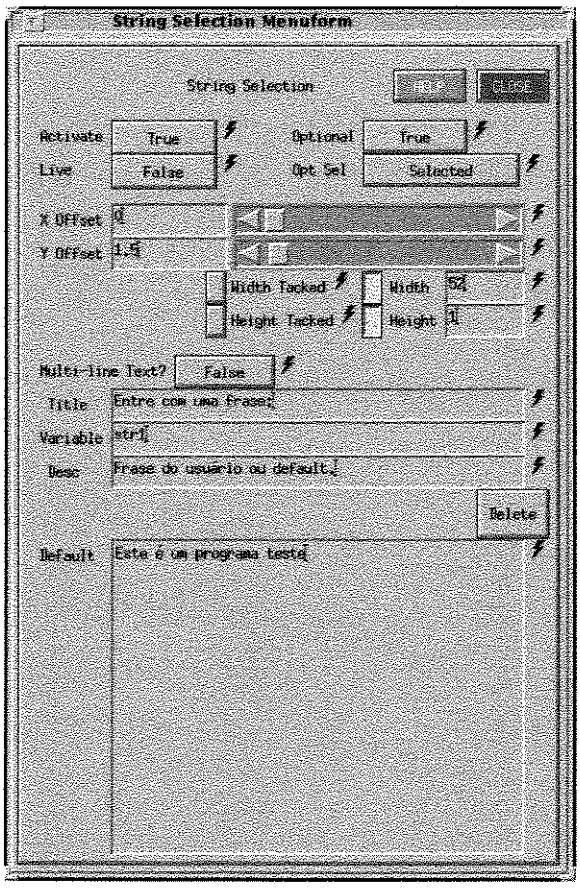

Figura 4.27: Janela da variável string com os campos alterados.

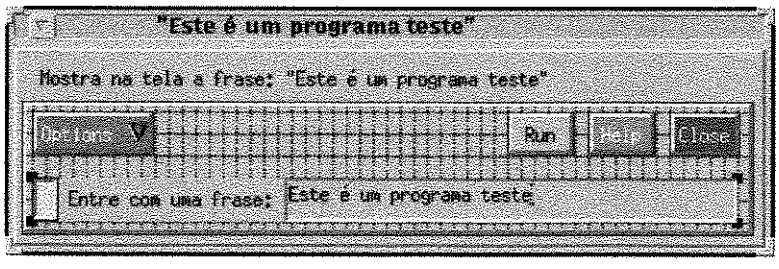

Figura 4.28: O pane da kroutine k_teste com a variável string já modificada.

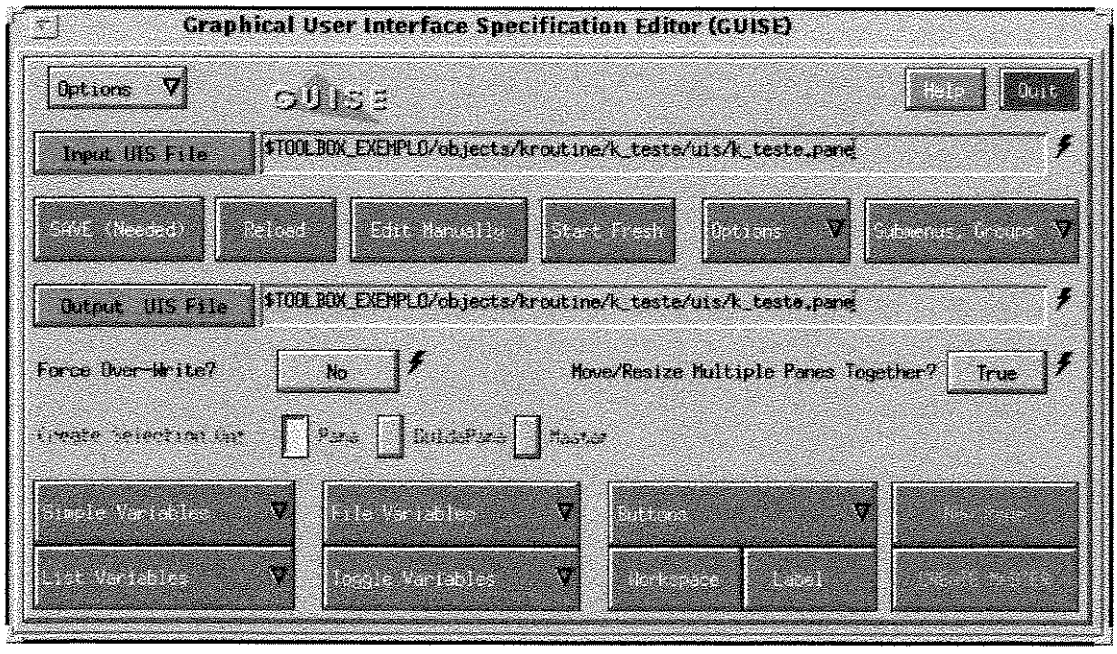

Figura 4.29: Janela do Guise antes do botão "SAVE(Needed)" ser pressionado para salvar as alterações feitas na interface gráfica de usuário.

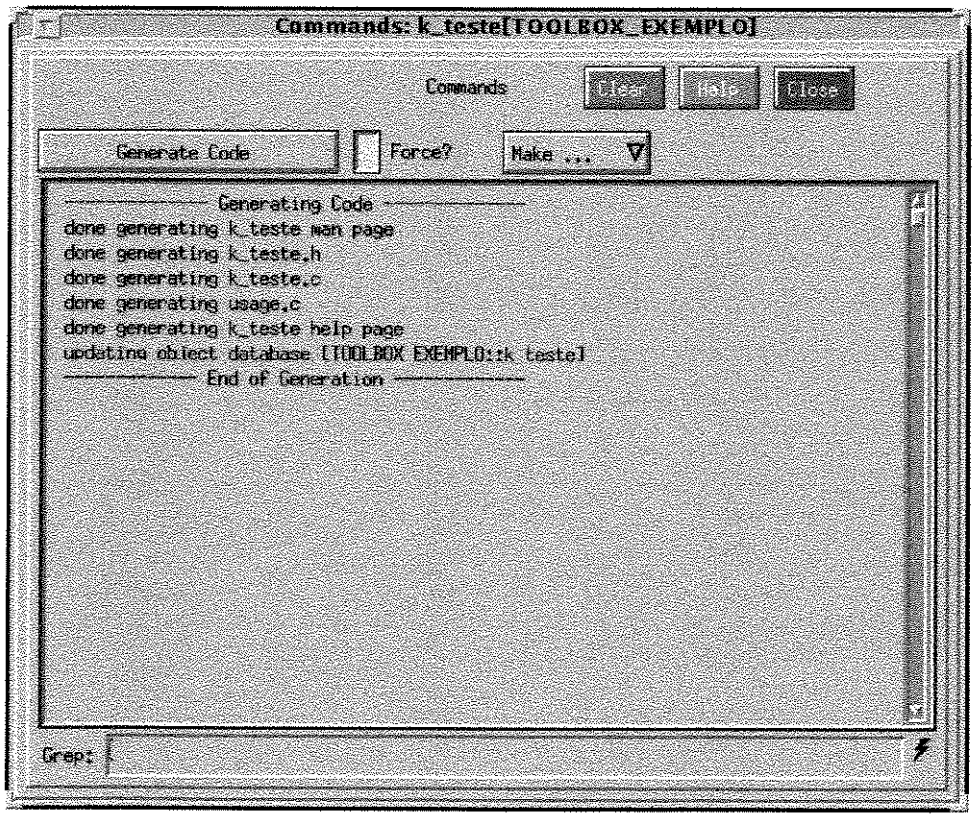

Figura 4.30: O subform "Commands" da janela do Composer já com o código gerado.

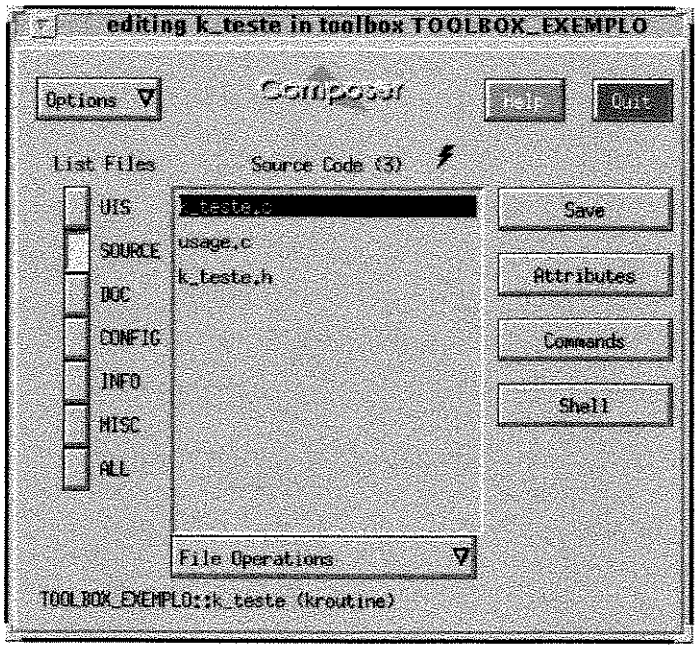

Figura 4.31: Janela do Composer com o botão "SOURCE" selecionado para o acesso aos programas fonte da kroutine k_teste.

tions" selecionando a opção "Edit" na janela do Composer. O arquivo k_teste.c gerado pelo Ghostwriter possui vários pares de tags onde será inserido o código em C. É importante que o texto inserido esteja entre estes pares de tags para que não se perca o código inserido quando o arquivo tiver que ser atualizado. A Figura 4.32 mostra parte do arquivo k_teste.c gerado pelo Ghostwriter antes da modificação do arquivo k_teste.pane.

A Figura 4.33 mostra parte do arquivo k_teste.c gerado pelo Ghostwriter depois das modificações do arquivo k_teste.pane. Para que o código da kroutine seja atualizado e tenha a aparência da figura anterior, é necessário que o código seja gerado novamente pressionando o botão "Generate Code" do subform "Commands" da janela do Composer.

A Figura 4.34 mostra parte do arquivo k_teste.c gerado pelo Ghostwriter depois das modificações necessárias para o seu funcionamento. Pode ser observado que o código inserido está entre o par de tags : /* -main_before_lib_call */ e /* -main_before_lib_call_end */.

No processo de desenvolvimento do software a documentação é um item importante. O Craftsman gera automaticamente a página help (k_teste.hlp) e a página man (k_teste.1) quando o programa (program object) é gerado. Para ver todos os arquivos de documenta-

```
/*-------------------------------------------------------

 Routine Name: main() - Mostra na tela a frase: "Este é um programa teste"

    Purpose: main program for k_teste

     Input:
                char *clui_info->i_file; {First Input data object}
                int  clui_info->i_flag; {TRUE if -i specified}

                char *clui_info->o_file; {Resulting output data object}
                int  clui_info->o_flag; {TRUE if -o specified}

    Output:
   Returns:

  Written By:
      Date: Sep 28, 1998
Modifications:

-------------------------------------------------------*/
```

Figura 4.32: Parte do arquivo k_teste.c gerado pelo Ghostwriter.

```
/*-------------------------------------------------------

 Routine Name: main() - Mostra na tela a frase: "Este é um programa teste"

    Purpose: main program for k_teste

     Input:
                char *clui_info->str1_string; {Frase do usuário ou default.}
                int  clui_info->str1_flag; {TRUE if -str1 specified}

    Output:
   Returns:

  Written By:
      Date: Sep 28, 1998
Modifications:

-------------------------------------------------------*/
```

Figura 4.33: Parte do arquivo k_teste.c gerado pelo Ghostwriter depois da modificação do arquivo k_teste.pane.

```
void main(
  int  argc,
  char **argv)
{

        kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
/* -main_variable_list_end */

        khoros_initialize(argc, argv, "TOOLBOX_EXEMPLO");
        kexit_handler(k_teste_free_args, NULL);

/* -main_get_args_call */
        kclui_initialize(PANEPATH, KGEN_KROUTINE, "TOOLBOX_EXEMPLO", "k_teste",
                k_teste_usage_additions, k_teste_get_args,
                k_teste_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */

if (clui_info->str1_string != NULL)
   kprintf("%s", clui_info->str1_string);
else kprintf("Este é um programa teste");

/* -main_before_lib_call_end */

/* -main_library_call */
/* -main_library_call_end */

/* -main_after_lib_call */
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}
```

Figura 4.34: Parte do arquivo k_teste.c com as modificações necessárias para seu funcionamento.

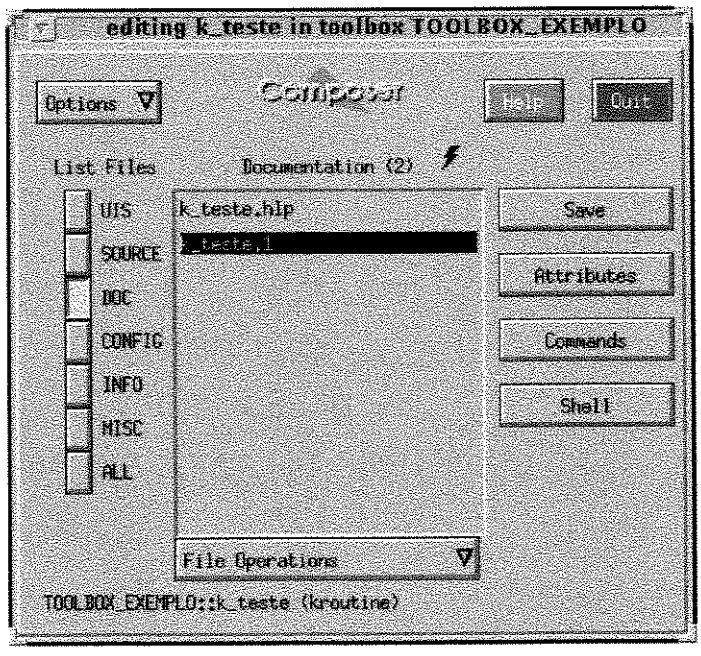

Figura 4.35: Janela do Composer com o botão "DOC" selecionado para acesso aos arquivos de documento.

ção no Composer, pressiona-se o botão "DOC" do lado esquerdo da janela do Composer. A Figura 4.35 mostra a janela do Composer com a opção "DOC" selecionada.

Todos os arquivos relacionados a opção "DOC" aparecerão no centro da janela do Composer. Para editar o arquivo k_teste.1, basta que ele esteja selecionado e então pressionar o botão "File Operations" da janela do Composer selecionando a opção "Edit". Aparecerá um editor de texto para que o arquivo k_teste.1 seja editado. Antes de editado, o arquivo deverá parecer-se com o da Figura 4.36.

O arquivo de documentação, como o código fonte, também possui tags em seu texto. Mas estes tags não são tão óbvios como os gerados pelo código fonte. Todas as linhas que aparerecem na figura anterior são geradas automaticamente e não devem ser alteradas.

Primeiramente edita-se o campo de descrição. Este campo está entre dois tags : .SH DESCRIPTION e .SH "REQUIRED ARGUMENTS". Como o código fonte, o texto colocado entre estes tags será preservado. O texto inserido para o programa k_teste no arquivo k_teste.1 é mostrado na Figura 4.37.

O campo de exemplos também pode ser editado . Pode-se colocar exemplos de como executar o programa k_teste pela linha de comando. Esses exemplos foram colocados

```
.TH "k_teste" "TOOLBOX_EXEMPLO" "COMMANDS" "" "Sep 28, 1998"
.SH PROGRAM
k_teste \- Mostra na tela a frase: "Este é um programa teste"
.syntax TOOLBOX_EXEMPLO k_teste
.SH DESCRIPTION
.SH "REQUIRED ARGUMENTS"
none
.sp
.SH "OPTIONAL ARGUMENTS"
.IP -str1 7
type: string
.br
desc: Frase do usuário ou default.
.br
default: Este é um programa teste
.br
.sp
.SH EXAMPLES
.SH "SEE ALSO"
.SH RESTRICTIONS
.SH REFERENCES
.SH COPYRIGHT
Copyright (C) 1993 - 1996, Khoral Research, Inc.  All rights reserved.
```

Figura 4.36: Arquivo k_teste.1 gerado pelo Ghostwriter.

```
.SH DESCRIPTION
.LP
Este programa mostra na tela uma frase especificada pelo usuário. Se ela não for
especificada, a seguinte frase aparecerá na tela: "Este é um programa teste".
.SH "REQUIRED ARGUMENTS"
```

Figura 4.37: Parte do arquivo k_teste.1 modificado.

entre tags como mostra a Figura 4.38.

Também podem ser editados os campos "SEE ALSO", "RESTRICTIONS" e "REFERENCES". Depois de todos os campos editados, o arquivo k_teste deve ser fechado. Então selecione a opção "Formatted" do menu "File Operations" da janela do Composer.

Para que o arquivo k_teste.hlp seja mudado é necessário que primeiro o arquivo k_teste.1 seja editado. Depois de editado o arquivo k_teste.1, para que as alterações no arquivo k_teste.hlp sejam feitas, pressione o botão "Commands" da janela do Composer para aparecer uma nova janela e nesta pressione o botão "Generate Code". Para ver a página help formatada basta pressionar o botão "File Operation" e selecionar a opção "Formatted".

```
.SH EXAMPLES
.LP
O exemplo seguinte mostra a frase "Este é um programa teste".
.sp
% K_teste
.br
O exemplo seguinte mostra a frase "Esta é a frase do usuário".
.br
% k_teste str1 "Esta é a frase do usuário"
.sp
.SH "SEE ALSO"
```

Figura 4.38: Parte do arquivo k_teste.1 modificado.

Compilar e testar o programa

Uma vez feitas todas as mudanças necessárias para o perfeito funcionamento do programa, este estará pronto para ser compilado. Deve-se lembrar de atualizar o código toda vez que forem feitas alterações nos arquivos *.pane, *.c, *.1 e outros antes que o programa seja compilado. Para esta atualização deve-se pressionar o botão "Commands" da janela do Composer para que abra a janela "Commands" e pressiona-se o botão "Generate Code". Para compilar o programa, basta selecionar do menu "Make" a opção "Make Install". Se nenhum erro for encontrado, a janela deverá parecer-se com a da Figura 4.39.

Depois do programa compilado, pode-se executar o programa da linha de comando. Para isso pode-se executar das duas maneiras seguintes :

% k_teste

% k_teste -str1 "Este é o lugar da frase do usuário"

Testando o novo programa no Cantata

Primeiro deve-se executar o Cantata da linha de comando da seguinte forma :

% cantata

Do menu "Glyphs" seleciona-se a Category "Toolbox_Exemplo" e a Subcategory "Programa teste" e finalmente o icon name Kroutine teste. Na área de trabalho do Cantata aparecerá o glyph igual ao da Figura 4.40.

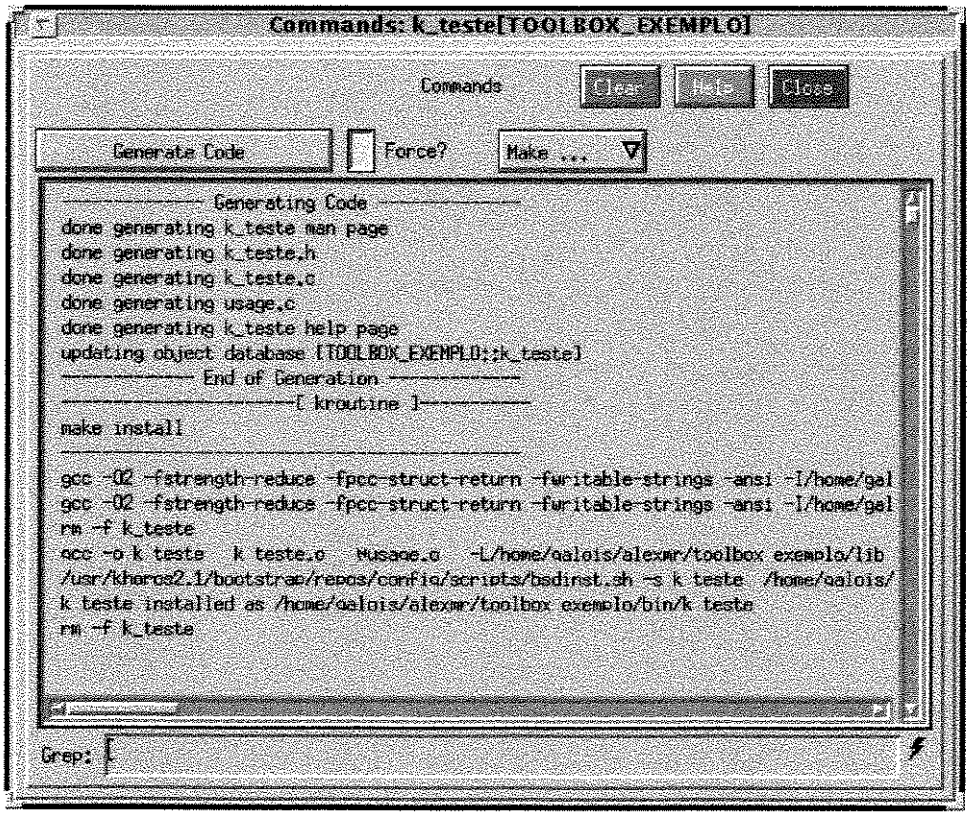

Figura 4.39: Janela "Commands" depois de compilado o programa k_teste.

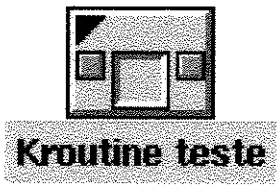

Figura 4.40: Glyph da Kroutine teste.

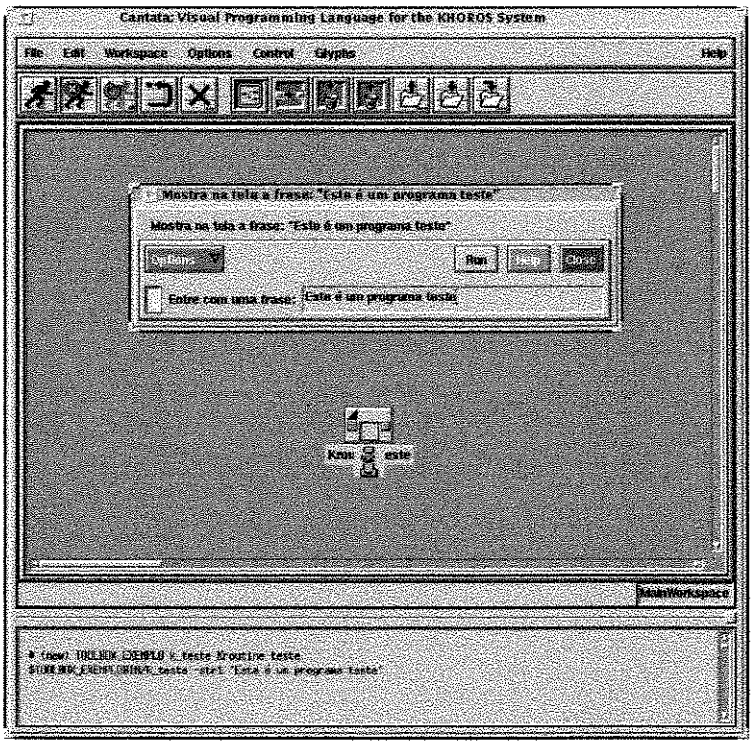

Figura 4.41: Janela do Cantata com o glyph Kroutine teste e sua interface gráfica de usuário.

Antes de executar o programa, deve-se verificar o pane do glyph. Para abrir o pane basta pressionar o triângulo preto no canto superior esquerdo do glyph. Para executar o programa, basta pressionar o botão do centro do glyph. Para visualizar o resultado da execução do programa, deve-se pressionar o prompt na forma de um "i" azul. Surgirá uma janela com o resultado do programa. Na Figura 4.41, pode-se vizualizar o pane com a frase "Este é um programa teste" e o prompt azul informando que o programa foi executado e que existe uma saída para ele. Esta saída pode ser vista na Figura 4.42.

Na Figura 4.43, pode-se vizualizar o pane com a frase "Frase do usuário!!!" e o prompt azul informando que o programa foi executado e que existe uma saída para ele. Esta saída pode ser vista na Figura 4.44. Na Figura 4.45 pode-se vizualizar o pane com a frase desativada e o prompt azul informando que o programa foi executado e que existe uma saída para ele. Esta saída pode ser vista na Figura 4.46.

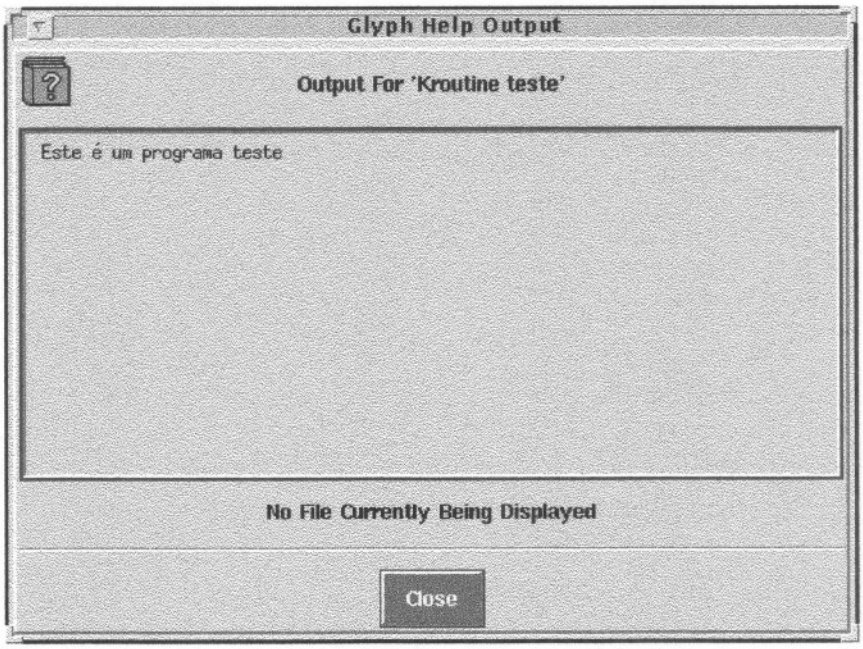

Figura 4.42: Resultado da execução do programa.

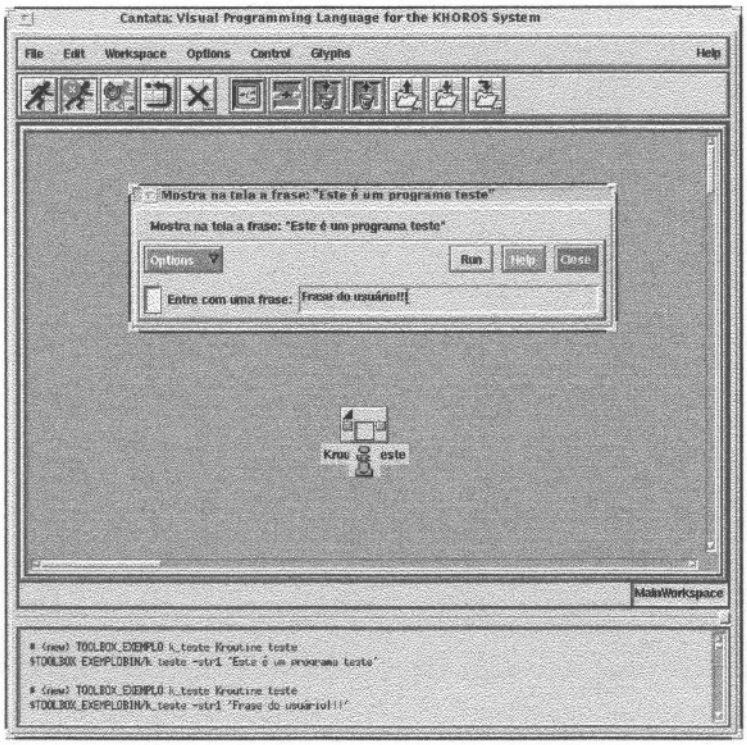

Figura 4.43: Janela do Cantata com o glyph Kroutine teste e sua interface gráfica de usuário.

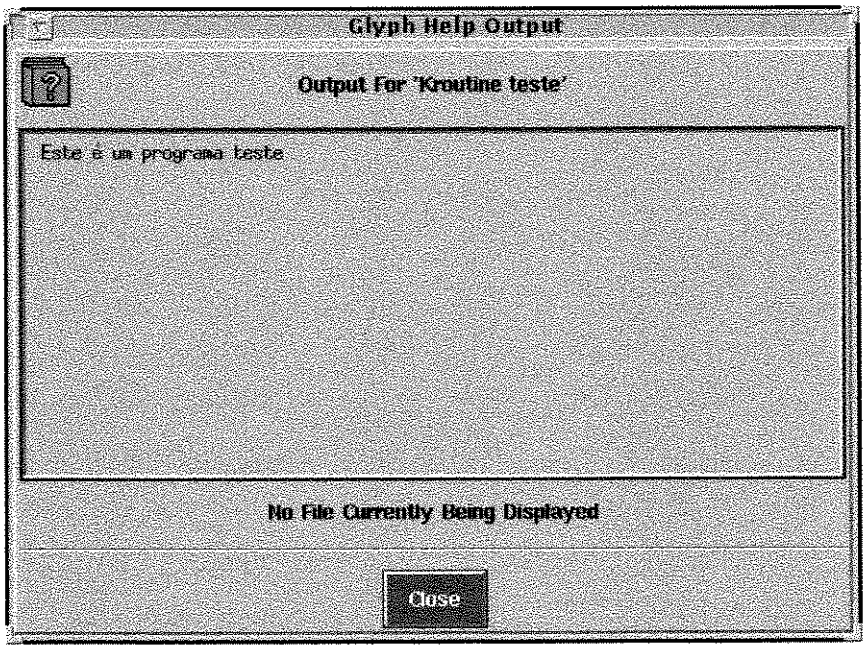

Figura 4.46: Resultado da execução do programa.

4.5.3. Exemplo de uma kroutine com uma função biblioteca associada.

Selecionar a toolbox para a kroutine

Deve-se seguir o mesmo procedimento adotado pelo programa k_teste.

Através do Craftsman criar a kroutine

Para criar o programa teste_lk, deverão ser adotados os procedimentos anteriores com algumas modificações. A opção "Yes" deverá ser selecionada no atributo "Create Library Routine ?" para que seja criado um arquivo de biblioteca associado ao programa. A janela com os campos preenchidos deverá parecer-se com a da Figura 4.47.

Depois de preenchidos todos os campos da janela, pressiona-se o botão "Create kroutine" para que o novo programa seja criado.

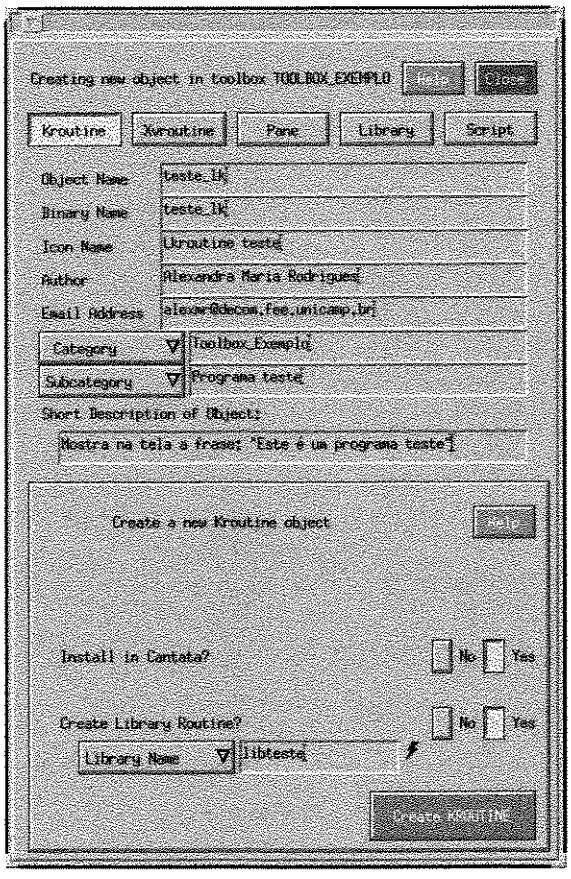

Figura 4.47: Janela para a criação de uma kroutina com os campos preenchidos.

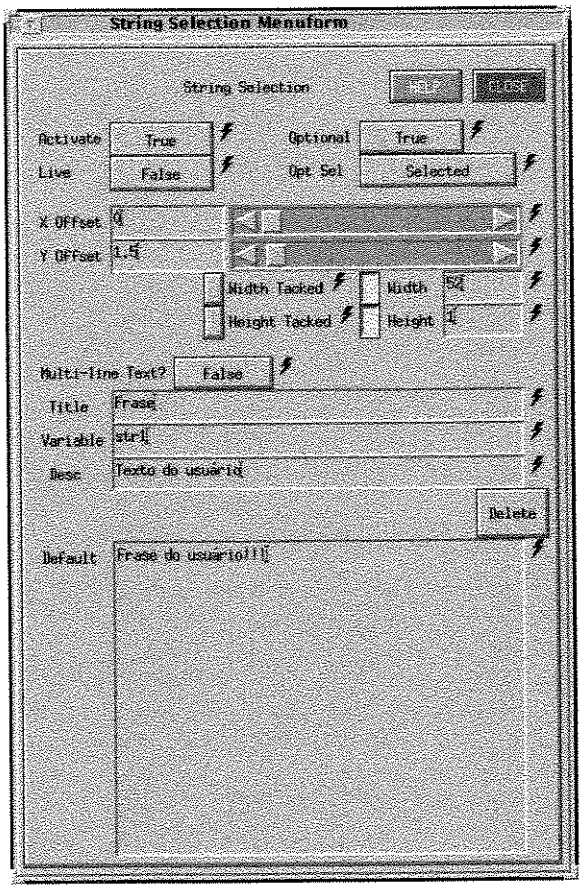

Figura 4.48: Janela dos atributos da variável string.

Editar a kroutine

Precisa-se editar o arquivo de interface para o usuário, o arquivo teste_lk.pane. Para isso, com o programa selecionado pressiona-se o botão "Object Operation" e escolhe-se a opção "Edit Object (Composer)".

Seleciona-se o botão "File Operations" e escolhe-se a opção "Guise" para a edição do arquivo teste_lk.pane, que deve ser igual ao arquivo k_teste.pane.

Para mudar os atributos da variável string, basta pressionar o botão do meio do mouse para aparecer a janela da variável string. Deve-se preencher os campos desta janela como mostrado na Figura 4.48.

Então a interface gráfica de usuário deverá ser parecida com a Figura 4.49.

Depois das alterações no arquivo teste_lk.pane, gera-se novamente o código usando o botão "Generate Code" do subform "Commands" da janela do Composer. A Figura 4.50

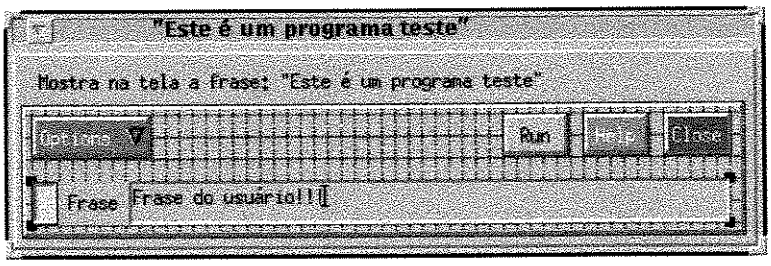

Figura 4.49: Interface gráfica de usuário, o pane, para a kroutine teste_lk.

mostra o subform "Commands" depois de pressionado o botão "Generate Code".

Feitas as alterações da interface gráfica de usuário, o arquivo fonte será editado. O acesso ao arquivo fonte é feito pressionando o botão "SRC" no lado esquerdo da janela do Composer. No centro da janela aparecerão todos os arquivos fontes associados ao programa teste_lk.

Para editar o arquivo teste_lk.c e o arquivo lteste_lk, deve-se selecionar um de cada vez e pressionar o botão "File Operations" selecionando a opção "Edit" na janela do Composer. É importante que o texto inserido esteja entre pares de tags para que não se perca o código inserido. A Figura 4.51 mostra parte do arquivo teste_lk.c gerado pelo Ghostwriter antes das modificações necessárias para o seu funcionamento. A Figura 4.52 mostra parte do arquivo teste_lk.c gerado pelo Ghostwriter depois das modificações necessárias para o seu funcionamento e a Figura 4.53 mostra o arquivo lteste_lk.c gerado pelo Ghostwriter antes das modificações necessárias para o seu funcionamento enquanto que a Figura 4.54 mostra parte do arquivo lteste_lk.c gerado pelo Ghostwriter depois das modificações necessárias para o seu funcionamento.

Compilar e testar o programa

Uma vez feitas todas as mudanças necessárias para o perfeito funcionamento do programa, este estará pronto para ser compilado. Deve-se lembrar de atualizar o código toda vez que forem feitas alterações nos arquivos *.pane, *.c, *.1 e outros antes que o programa seja compilado. Para esta atualização, deve-se pressionar o botão "Commands" da janela do Composer para que abra a janela "Commands" e pressiona-se o botão "Generate Code". Primeiro, deve-se complilar o arquivo de biblioteca. Para isso seleciona-se a opção

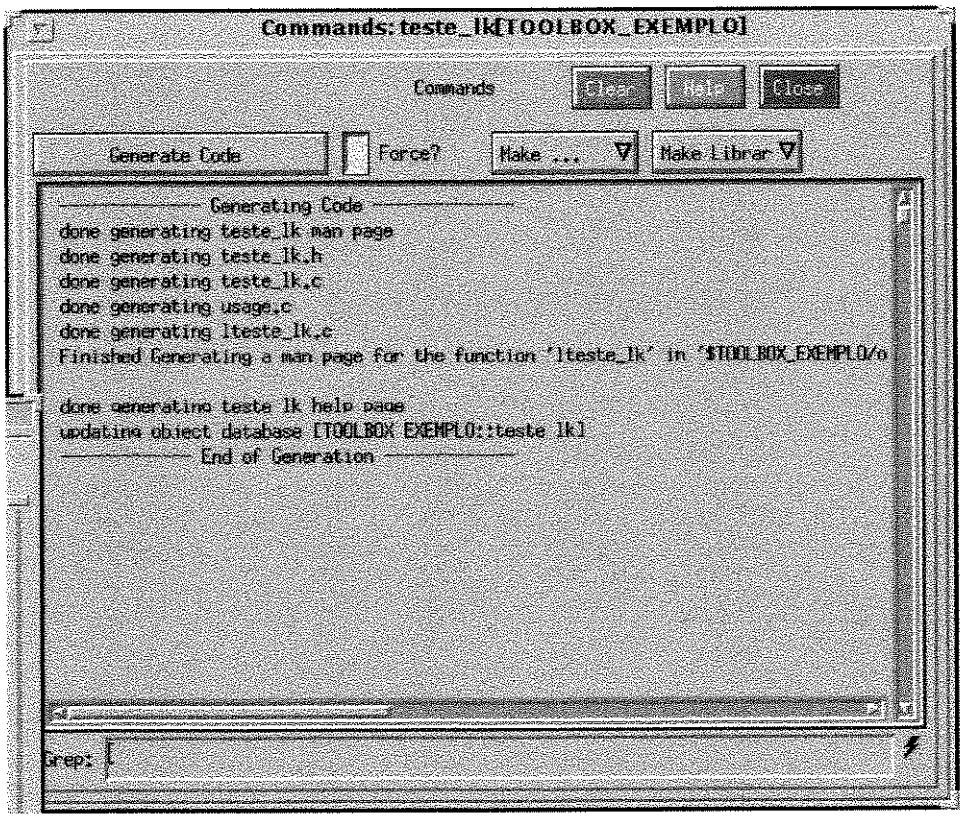

Figura 4.50: Subform "Commands" com o código já gerado.

```
void main(
  int argc,
  char **argv)
{

        kform *pane;        /* form tree representing *.pane file */
/* -main_variable_list */
/* -main_variable_list_end */

        khoros_initialize(argc, argv, "TOOLBOX_EXEMPLO");
        kexit_handler(teste_lk_free_args, NULL);

/* -main_get_args_call */
        kclui_initialize(PANEPATH, KGEN_KROUTINE, "TOOLBOX_EXEMPLO",
"teste_lk",
                teste_lk_usage_additions, teste_lk_get_args,
                teste_lk_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
/* -main_before_lib_call_end */

/* -main_library_call */
        lteste_lk();
/* -main_library_call_end */

/* -main_after_lib_call */
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}
```

Figura 4.51: Parte do arquivo teste_lk gerado pelo Ghostwriter.

```
void main(
  int  argc,
  char **argv)
{

        kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
/* -main_variable_list_end */

        khoros_initialize(argc, argv, "TOOLBOX_EXEMPLO");
        kexit_handler(teste_lk_free_args, NULL);

/* -main_get_args_call */
        kclui_initialize(PANEPATH, KGEN_KROUTINE, "TOOLBOX_EXEMPLO", "teste_lk",
                teste_lk_usage_additions, teste_lk_get_args,
                teste_lk_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
/* -main_before_lib_call_end */

/* -main_library_call */
        lteste_lk(clui_info->str1_string);
/* -main_library_call_end */

/* -main_after_lib_call */
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}
```

Figura 4.52: Parte do arquivo teste_lk.c já editado.

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>> Library Routine for teste_lk
  >>>>
  >>>> Private:
  >>>>
  >>>>  Static:
  >>>>  Public:
  >>>>  lteste_lk
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "internals.h"

/* -library_includes */
/* -library_includes_end */


/*****************************************************************
*
* Routine Name: lteste_lk - short description of lteste_lk()
*
*     Purpose: This should be a complete description that anyone
*              could understand;  it should have acceptable grammar
*              and correct spelling.
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*     Returns: TRUE (1) on success, FALSE (0) otherwise
*
* Restrictions: Restrictions on data or input as applicable
*   Written By:
*         Date: Sep 28, 1998
*     Verified:
*  Side Effects:
*     Examples:
* Modifications:
*****************************************************************/
/* -library_def */
int lteste_lk(void)
/* -library_def_end */

/* -library_code */
{
        return TRUE;
}
/* -library_code_end */
```

Figura 4.53: Arquivo lteste_lk.c gerado pelo Ghostwriter.

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>> Library Routine for teste_lk
  >>>>
  >>>> Private:
  >>>>
  >>>>  Static:
  >>>>  Public:
  >>>> lteste_lk
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "internals.h"

/* -library_includes */
/* -library_includes_end */


/*****************************************************************
*
* Routine Name: lteste_lk - short description of lteste_lk()
*
*     Purpose: This should be a complete description that anyone
*              could understand;  it should have acceptable grammar
*              and correct spelling.
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*     Returns: TRUE (1) on success, FALSE (0) otherwise
*
* Restrictions: Restrictions on data or input as applicable
*    Written By:
*         Date: Sep 28, 1998
*     Verified:
* Side Effects:
*     Examples:
* Modifications:
*****************************************************************/
/* -library_def */
int lteste_lk(char *string)
/* -library_def_end */

/* -library_code */
{
        if (string != NULL)
           kprintf("%s", string);
        else kprintf("Este é um programa teste");
        return TRUE;
}
/* -library_code_end */
```

Figura 4.54: Arquivo lteste_lk.c depois de editado.

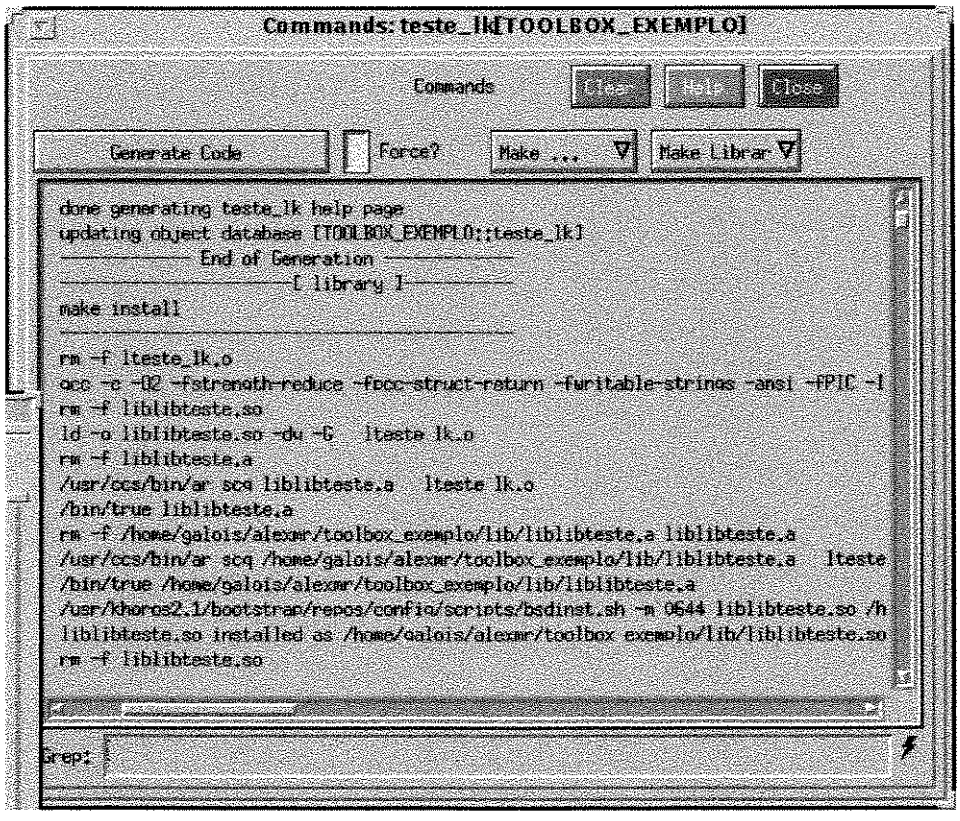

Figura 4.55: Subform "Commands" depois de compilado o arquivo de biblioteca.

"Make Install" do menu "Make Librar". Se nenhum erro for encontrado, a janela deverá parecer-se com a da Figura 4.55.

Para compilar o programa, basta selecionar do menu "Make..." a opção "Make Install". Se nenhum erro for encontrado a janela deverá parecer-se com a da Figura 4.56.

Testando o novo programa no Cantata

Primeiro deve-se executar o Cantata da linha de comando da seguinte forma :

% cantata

Do menu "Glyphs" seleciona-se a Category "Toolbox_Exemplo" e a Subcategory "Programa teste" e finalmente o icon name Lkroutine teste. Na área de trabalho do Cantata aparecerá o glyph igual ao da Figura 4.57.

Antes de executar o programa, deve-se verificar o pane do glyph. Para abrir o pane

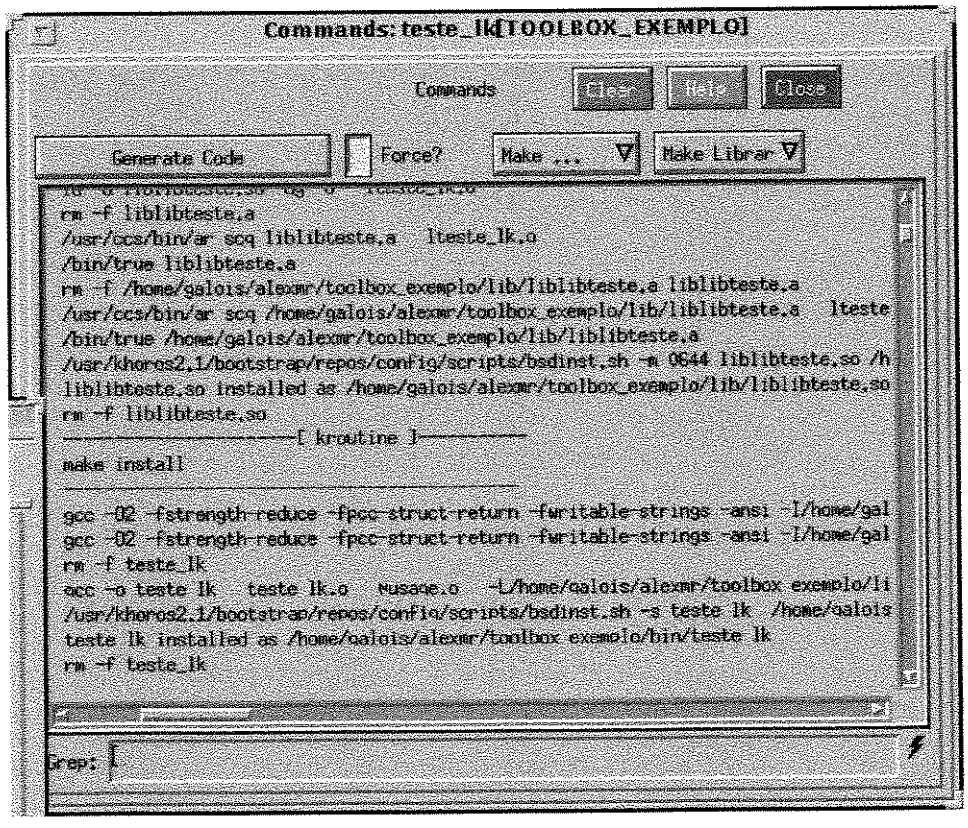

Figura 4.56: Subform "Commands" depois de compilado o programa.

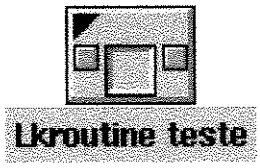

Figura 4.57: Glyph do programa teste_lk.

pressiona-se o triângulo preto no canto superior esquerdo do glyph e para executar o programa pressiona-se o botão do centro do glyph. Para visualizar o resultado da execução do programa, deve-se pressionar o prompt na forma de um "i" azul. Surgirá uma janela com o resultado do programa. Na Figura 4.58, pode-se vizualizar o pane com a frase "Frase do usuário!!!" e o prompt azul informando que o programa foi executado e que existe uma saída para ele. Esta saída pode ser vista na Figura 4.59.

Na Figura 4.60, pode-se vizualizar o pane com a frase desativada e o prompt azul informando que o programa foi executado e que existe uma saída para ele. Esta saída pode ser vista na Figura 4.61.

4.6. Pane objects

O pane object depende de outro software object, kroutine ou xvroutine, para funcionar. Uma vez criado um pane object, ele é apenas uma outra alternativa de interface de usuário para outro programa.

Os passos para o desenvolvimento de um pane object são :

1. Selecionar ou criar uma toolbox para o pane object.
2. Na toolbox selecionada ou criada, através do Craftsman criar um pane object.
   (a) Selecionar a toolbox desejada para o pane object da lista de "Toolboxes".
   (b) Selecionar a opção "Create Object" do menu "Object Operations" para que apareça o subform "Create new object".
   (c) Preencher os campos do subform "Create new object" e pressionar o botão "Create PANE".

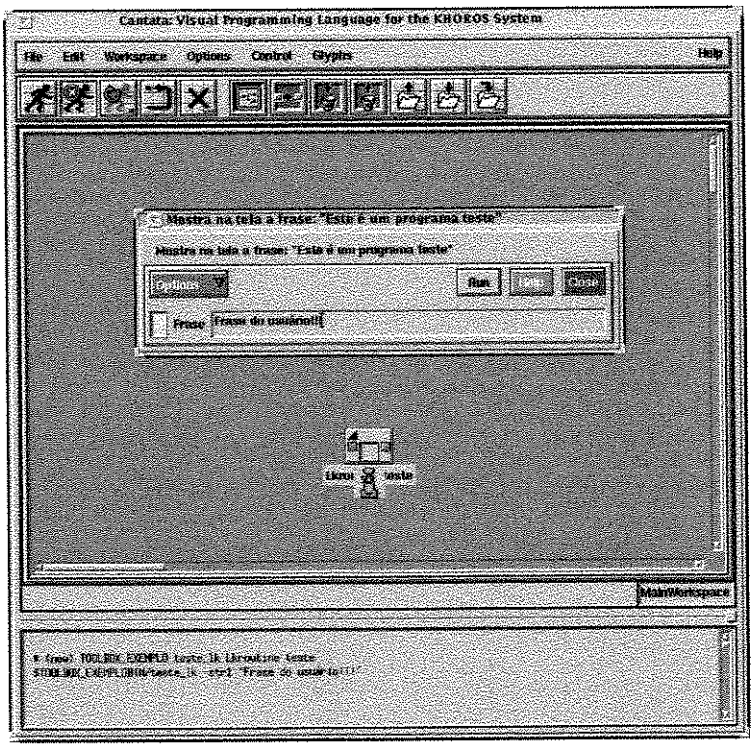

Figura 4.58: Janela do Cantata com o glyph Lkroutine teste e seu pane.

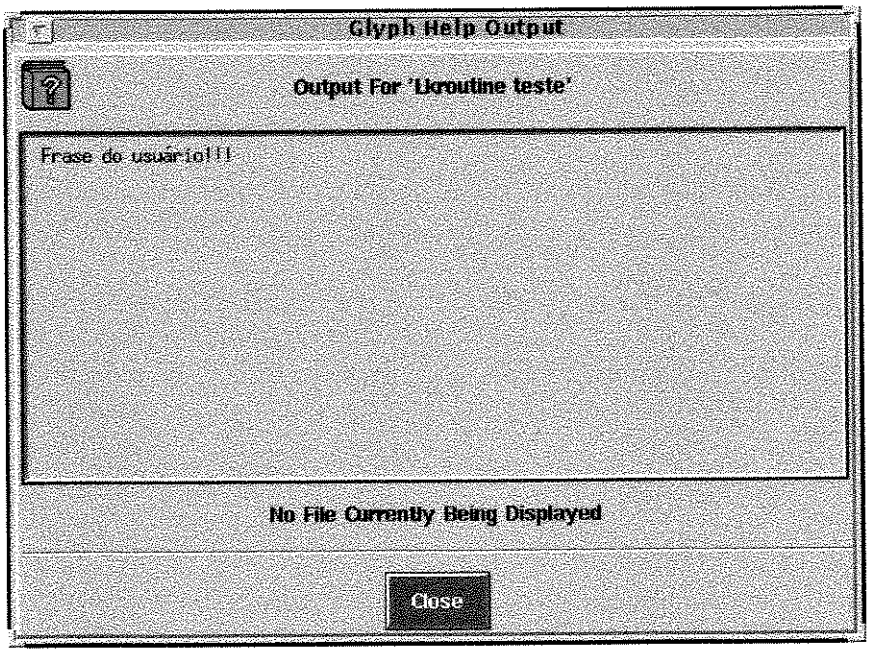

Figura 4.59: Resultado da execução do programa teste_lk.

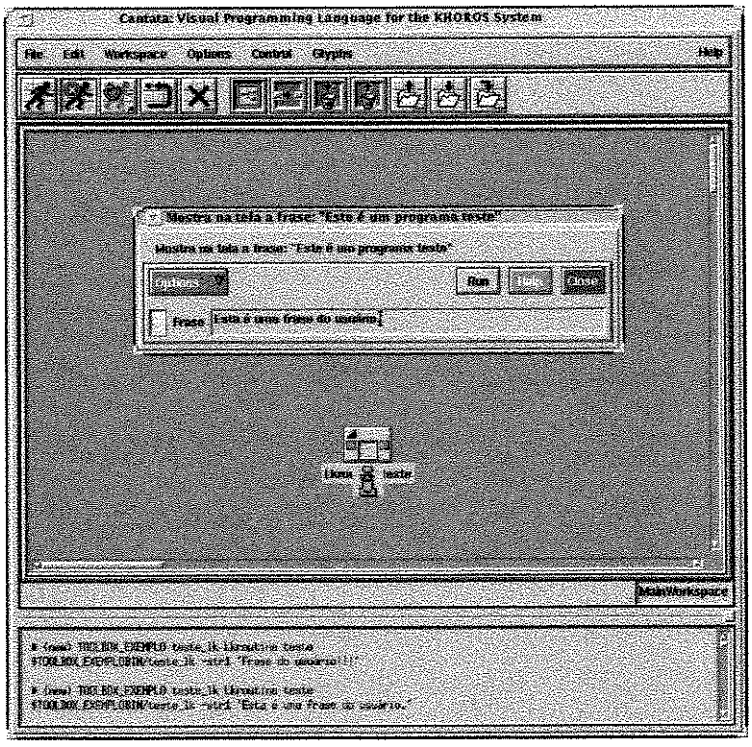

Figura 4.60: Janela do Cantata com o glyph Lkroutine e seu pane.

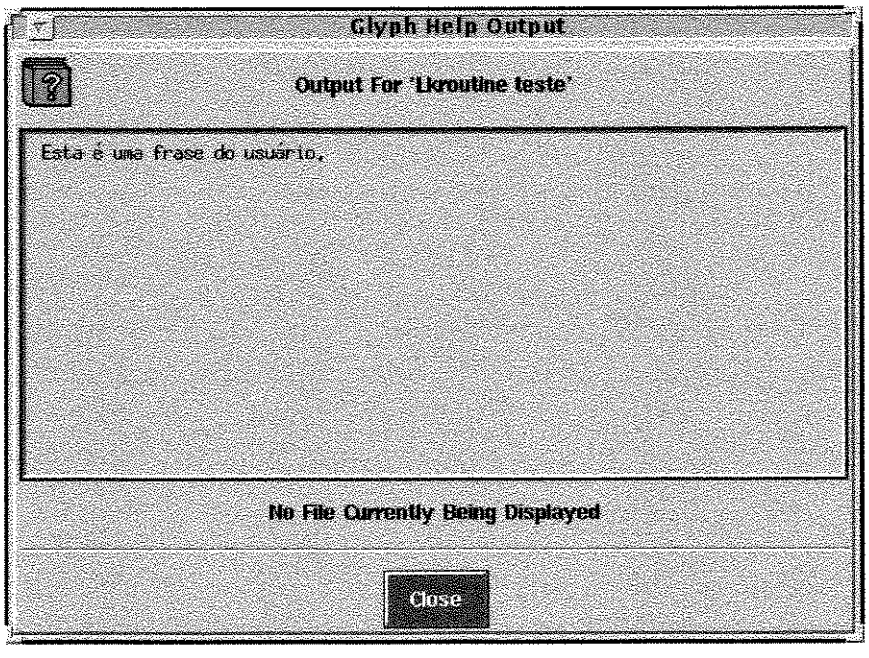

Figura 4.61: Resultado da execução do programa teste_lk.

3. Editar o pane object.
   (a) Selecionar o pane object criado da lista de software object da janela do Craftsman.
   (b) Selecionar a opção "Edit Object (Composer)" do menu "Object Operations" para que apareça a janela do Composer.
   (c) Da janela do Composer, pressionar o botão "UIS" e selecionar o arquivo *.pane.
   (d) Selecionar a opção "Guise" do menu "File Operations".
   (e) Especificar os argumentos da linha de comando do pane no arquivo *.pane.
4. Através do Composer, compilar e instalar o pane object.
   (a) Selecionar o subform "Commands". Pressionar o botão "Generate Code" para que o código do pane object seja atualizado.
   (b) Do subform "Commands" selecione a opção "Make Install" do menu "Make..."
5. Escrever uma página man. Editar esta página e através do Ghostwriter gerar automaticamente a página help.
6. Executar o pane object da linha de comando e verificar os resultados.
7. Executar o pane object dentro do Cantata.

4.7. Script objects

Segue-se os seguintes passos para a criação de um script :

1. Selecionar ou criar uma toolbox para o script.
2. Na toolbox selecionada ou criada, através do Craftsman criar um script object.
3. Editar o arquivo gerado pelo Ghostwriter, inserindo texto somente entre os pares de tags.

O gerador de código Ghostwriter gera um arquivo *.1 para o script no diretório man. Todas as adições neste arquivo devem ser feitas entre os pares de tags. A Tabela 4.1 mostra os pares de tags existentes num arquivo *.1. E mostra também entre quais pares de tags um certo tipo de texto deve ser inserido.

Pares de tags gerados pelo Ghostwriter para o arquivo *.1		
Tipo de texto	**Início do tag**	**Fim do tag**
Descrição pequena	binary_name \-	.sintax TOOL-BOX_NAME script_object_name
Descrição longa	.SH DESCRITION	.SH "REQUIRED ARGUMENTS"
Exemplos	.SH EXAMPLES	.SH "SEE ALSO"
Menção de outros programas	.SH "SEE ALSO"	.SH RESTRICTIONS
Restrições	.SH RESTRICTIONS	.SH REFERENCES
Referências	.SH REFERENCES	.SH COPYRIGHT

Tabela 4.1: Pares de tags para o arquivo *.1

A página help para o script é uma versão da página man com uma formatação diferente. Somente será gerada se o script usa um arquivo *.pane. Dentro do Cantata, esta página será mostrada quando o botão "Help" da interface gráfica de usuário do script é pressionado. A página help é gerada automaticamente pelo Ghostwriter baseada na página man. A página help nunca deve ser editada manualmente. Todas adições devem ser feitas na página man. O Ghostwriter faz automaticamente as mudanças na página help.

4.8. Conclusão

O capítulo tenta de uma forma simples, através de alguns exemplos, mostrar passos para o desenvolvimento de programas no ambiente Khoros. Junto com os passos, passa-se também a idéia do propósito de cada arquivo associado com os diferentes tipos de programa. Para cada tipo de programa que são as kroutines,xvroutines, library, pane e script faz-se um pequeno roteiro de passos para a criação desses programas.

Capítulo 5

Conclusão

Este trabalho aborda os procedimentos para o desenvolvimento de programas no ambiente já existente de domínio público, o Khoros. O Khoros é um ambiente de integração e desenvolvimento de software que enfatiza o processamento da informação e visualização dos dados[1]. Este ambiente foi desenvolvido pelo "The Khoros Group, Departament of Electrical and Computer Engineering, University of New Mexico, Albuquerque".

Um programa para o ambiente Khoros está dividido em 4 categorias : os program objects, os library objects, os pane objects e os script objects, sendo que os program objects se dividem em kroutines e xvroutines. Um programa para o ambiente Khoros, independente da categoria a que pertença, pode ser criado utilizando um conjunto de ferramentas fornecidas pelo próprio Khoros. Usando as próprias ferramentas do Khoros torna mais fácil o desenvolvimento de programas e garante que todos programas possam usar a biblioteca Khoros para manipulação de dados. As ferramentas para adição de um programa no ambiente Khoros são : Craftsman, Composer e Guise.

Apresentou-se, assim, inicialmente, um estudo resumido sobre o Cantata, ferramenta de programação visual do sistema Khoros, uma vez que é neste ambiente que simulações serão realizadas e que programas serão executados através de uma programação visual com fluxo de dados. As ferramentas Craftsman, Composer e Guise foram estudadas visando a inserção de programas no ambiente Khoros. Sugeriu-se procedimentos para a inserção de programas no ambiente Khoros e o propósito de cada arquivo associado com cada tipo de programa, assim como várias questões relacionadas ao desenvolvimento de código usando o Khoros. Foi também apresentado um simples programa em linguagem C para tornar

fácil a compreensão de como os programas são inseridos no ambiente Khoros.

A seguir, são apresentados alguns dos programas desenvolvidos para o grupo PDI/DECOM/UNICAMP e que já estão inseridos no ambiente Khoros.

- 8 programas principais
  - rgb_yuv.c
  - rgb_yiq.c
  - rgbycrcb.c
  - rgbyprpb.c
  - yuv_rgb.c
  - yiq_rgb.c
  - ycrcbrgb.c
  - yprpbrgb.c
- 8 programas auxiliares
  - abre_arq.c
  - r_w_rec.c
  - r_w_mat.c
  - modula.c
  - ler.c
  - aloc_mat.c
  - manipmat.c
  - filtros.c
- 4 programas cabeçalho
  - maniparq.h
  - macrovec.h
  - macromat.h
  - filtros.h

Conclui-se que os procedimentos sugeridos estão a altura de proporcionar o desenvolvimento de trabalhos de pesquisa. Este trabalho contribui para :

- um enriquecimento de possibilidades quanto à programação necessária para implementação de modelos de simulação para esquemas envolvendo sistemas de televisão digital.
- ser mais uma forma de interação entre os elementos do grupo de PDI/DECOM, proporcionando o desenvolvimento de um ambiente de programação cada vez melhor para se realizar simulações envolvendo sistemas de televisão digital.
- ser também uma orientação para inserção de programas escritos em linguagem C, que não envolvem televisão digital, no ambiente Khoros.

Apêndice A

Conversão de formatos de sinais de TV e ATV

A necessidade da redução da taxa de bits para transmissão de sinais de televisão digitalizados é devido a grande capacidade de canal requerida para a recuperação de uma imagem de alto padrão de qualidade, tornando-se assim uma das limitações para o uso de sinais de televisão digital. A maioria dos sistemas propostos pelos grandes centros de pesquisa na área utilizam os sinais na forma de luminância e crominância ou sinal composto, para o processamento de redução de taxa. Este apêndice apresenta programas desenvolvidos em linguagem C pelo colega Ayres M. A. do Nascimento para a composição e decomposição dos sinais para televisão, para o formato necessário e requerido pelo usuário, conforme o sistema PAL-M, NTSC ou HDTV na forma de componentes de luminância e crominância ou sinal composto. Os dados utilizados para as várias composições estão de acordo com as recomendações do CCIR 1990, BROADCASTING SERVICE (TELEVISION).

Os programas estão divididos em : programas principais, programas auxiliares e programas cabeçalhos. Os programas principais são os desenvolvidos com o objetivo da geração de algum tipo de sinal, contendo a função main() e portanto podem ser transformados em programas executáveis. Os programas auxiliares são programas que contém as várias funções que são utilizadas pelos programas principais para a realização e composição dos sinais. Os programas cabeçalhos são aqueles que contém as definições utilizadas nos demais programas inclusive macros e portanto, seus nomes são seguidos pela extensão ".h".

As macros são aquelas funções dedicadas que realizam tarefas específicas para outros programas.

Os programas principais para a geração de sinais são os seguintes :

RGB_YUV.C	YUV_RGB.C
RGB_YIQ.C	YIQ_RGB.C
RGBYCrCb.C	YCrCbRGB.C
RGBYPrPb.C	YPrPbRGB.C

Os programas auxiliares são os seguintes :

ABRE_ARQ.C	LER.C
R_W_REC.C	ALOC_MAT.C
R_W_MAT.C	MANIPMAT.C
MODULA.C	FILTROS.C

Os programas cabeçalho são os seguintes :

MANIPARQ.H	MACROMAT.H
MACROVEC.H	FILTROS.H

Algumas funções usadas nos programas principais são fornecidas pelo Khoros :

kpds_open_output_object()	Abre um objeto de saída para escrita.
kpds_close_object()	Fecha um objeto de dado.
kpds_set_attribute()	Fixa os valores de um atributo num objeto de dado.
kpds_put_data()	Armazena uma unidade de dado em um segmento no objeto de dado.

Os glyphs dos programas inseridos no ambiente Khoros são apresentados na Figura A.1.

Imagem original usada nos programas mencionados anteriormente é mostrada na Figura A.2.

Um exemplo de um workspace para o glyph RGB_YUV é mostrada na Figura A.3.

A interface gráfica de usuário do programa RGB_YUV é mostrada na Figura A.4.

A entrada da componente R da imagem original é mostrada na Figura A.5.

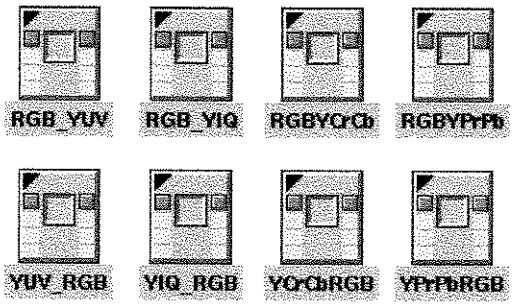

Figura A.1: Glyphs dos programas inseridos no Khoros.

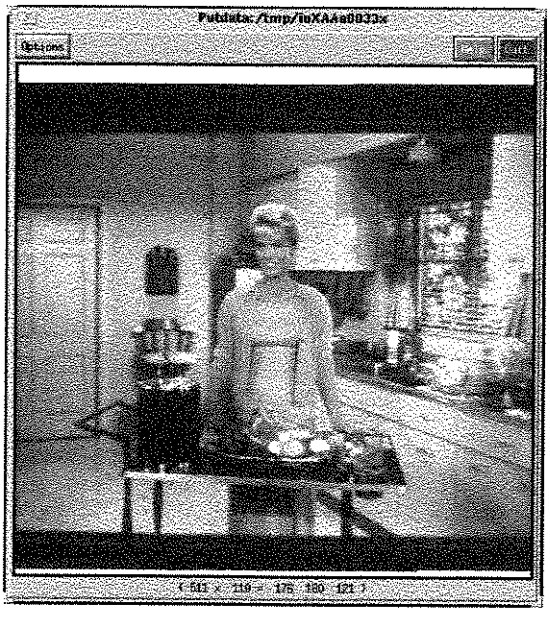

Figura A.2: Imagem original.

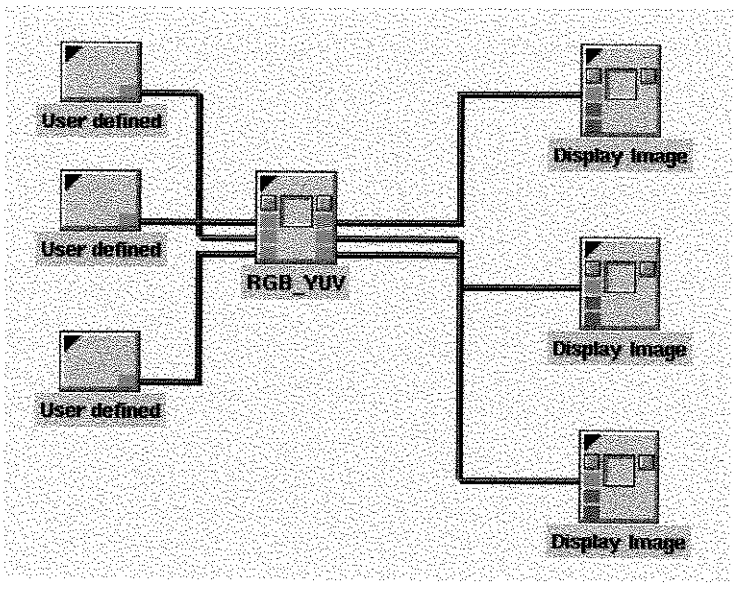

Figura A.3: Exemplo de um workspace para o glyph RGB_YUV.

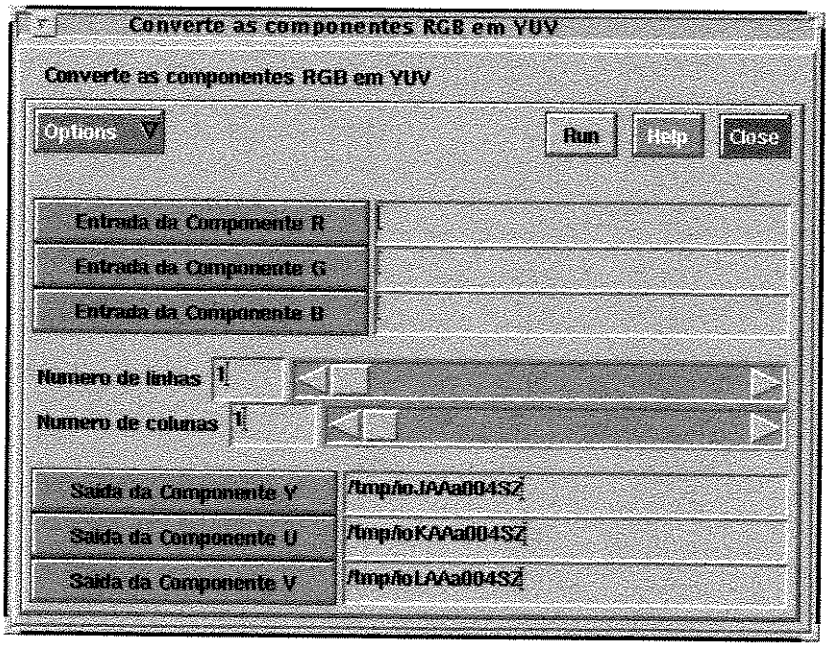

Figura A.4: Interface gráfica de usuário, pane, da kroutine rgb_yuv.

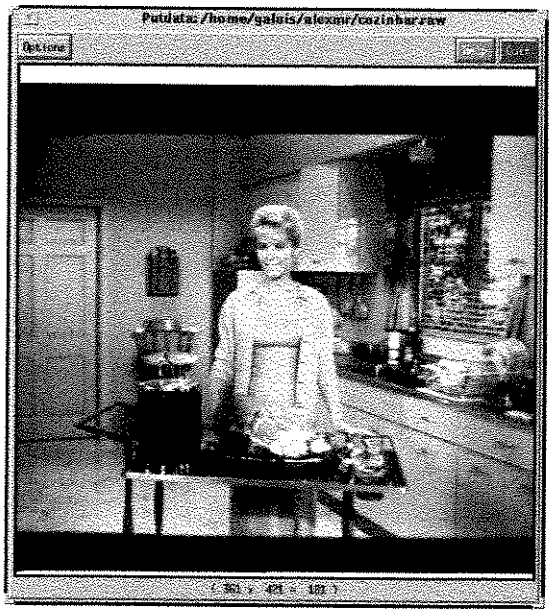

Figura A.5: Componente R da imagem original.

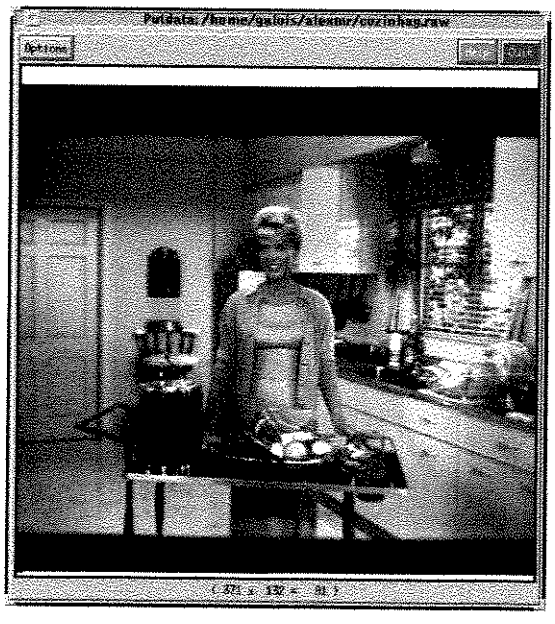

Figura A.6: Componente G da imagem original.

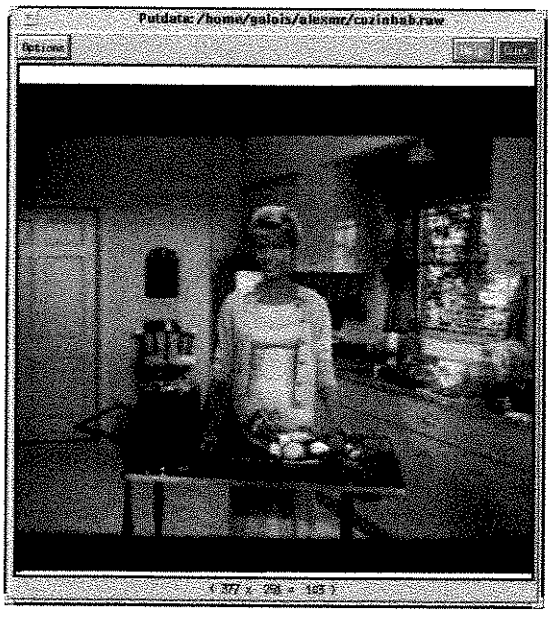

Figura A.7: Componente B da imagem original.

A entrada da componente G da imagem original é mostrada na Figura A.6.

A entrada da componente G da imagem original é mostrada na Figura A.7.

A saída da componente Y obtida pelo programa RGB_YUV é mostrada na Figura A.8.

A saída da componente U obtida pelo programa RGB_YUV é mostrada na Figura A.9.

A saída da componente V obtida pelo programa RGB_YUV é mostrada na Figura A.10.

Um exemplo de um workspace para o glyph RGB_YIQ é mostrada na Figura A.11.

A interface gráfica de usuário do programa RGB_YIQ é mostrada na Figura A.12.

As saídas das componentes I e Q obtidas pelo programa RGB_YIQ são mostradas respectivamente na Figura A.13 e na Figura A.14.

A interface gráfica de usuário do programa RGB_YCrCb é mostrada na Figura A.15.

As saídas das componentes Cr e Cb obtidas pelo programa RGBYCrCb são mostradas respectivamente na Figura A.16 e na Figura A.17.

A interface gráfica de usuário do programa RGB_YPrPb é mostrada na Figura A.18.

As saídas das componentes Pr e Pb obtidas pelo programa RGBYPrPb são mostradas respectivamente na Figura A.19 e na Figura A.20.

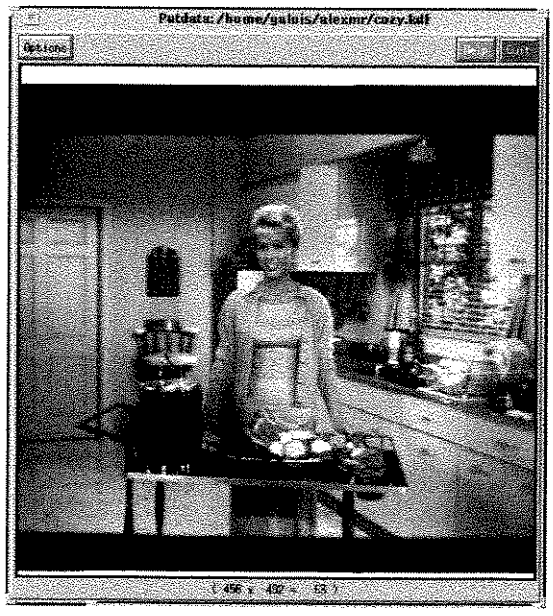

Figura A.8: Componente Y obtida na saída do programa RGB_YUV.

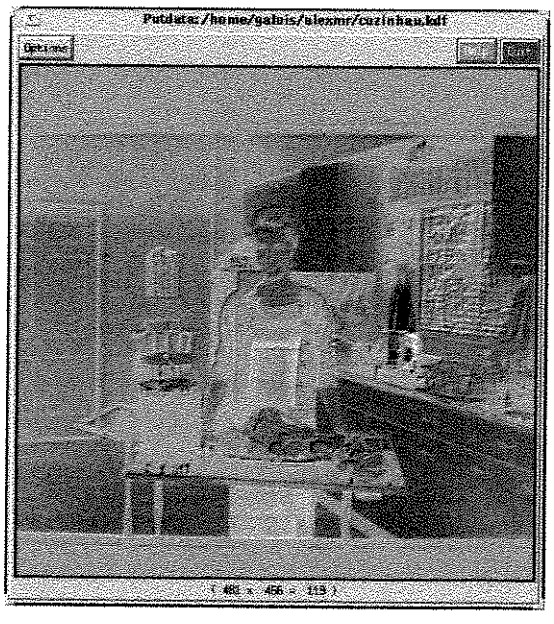

Figura A.9: Componente U obtida na saída no programa RGB_YUV.

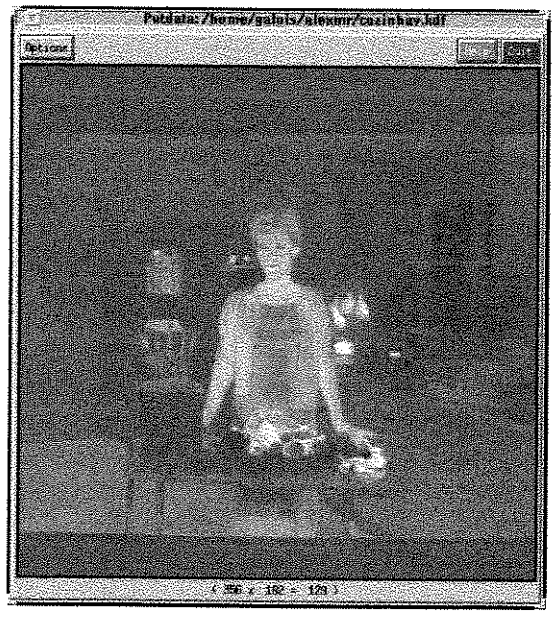

Figura A.10: Componente V obtida na saída do programa RGB_YUV.

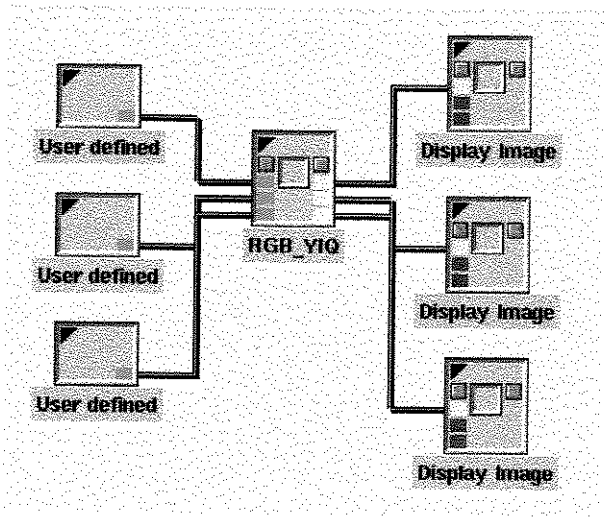

Figura A.11: Exemplo de workspace usando o glyph RGB_YIQ.

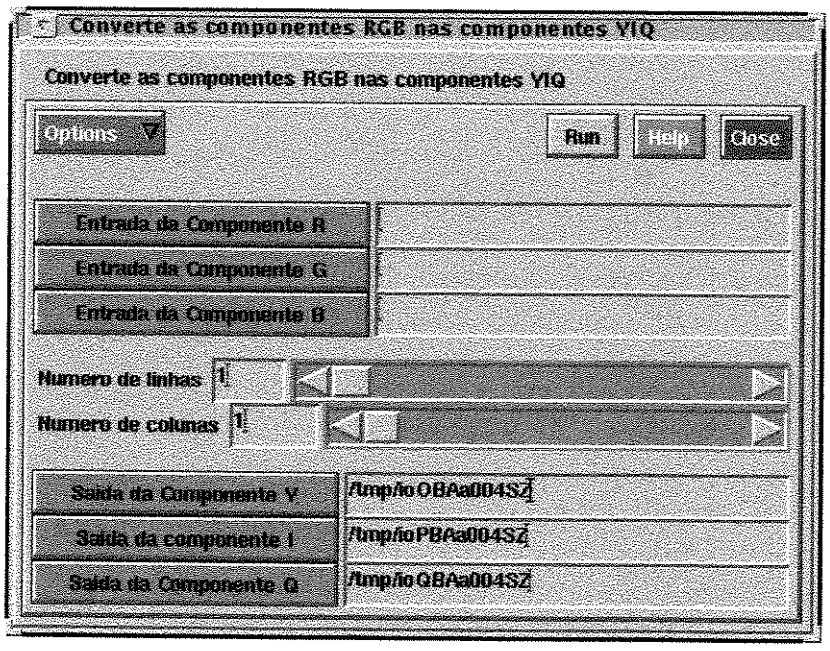

Figura A.12: Interface gráfica de usuário, pane, da kroutine rgb_yiq.

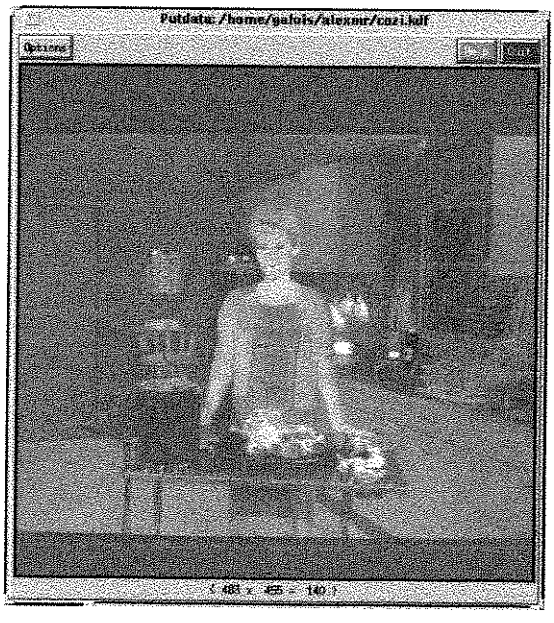

Figura A.13: Componente I obtida na saída do programa RGB_YIQ.

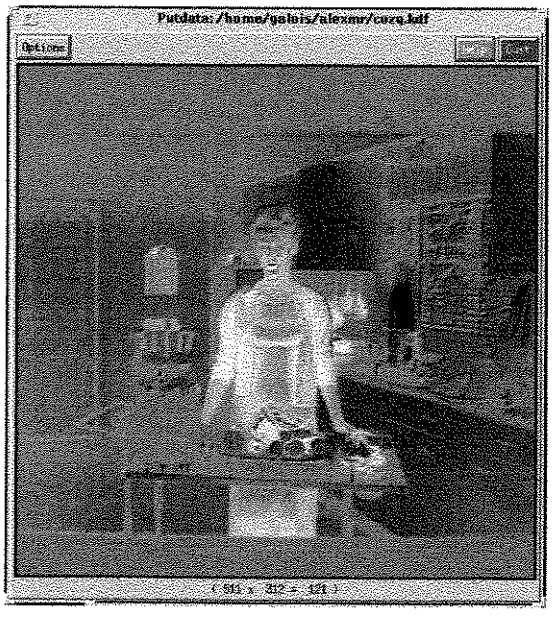

Figura A.14: Componente Q obtida na saída do programa RGB_YIQ.

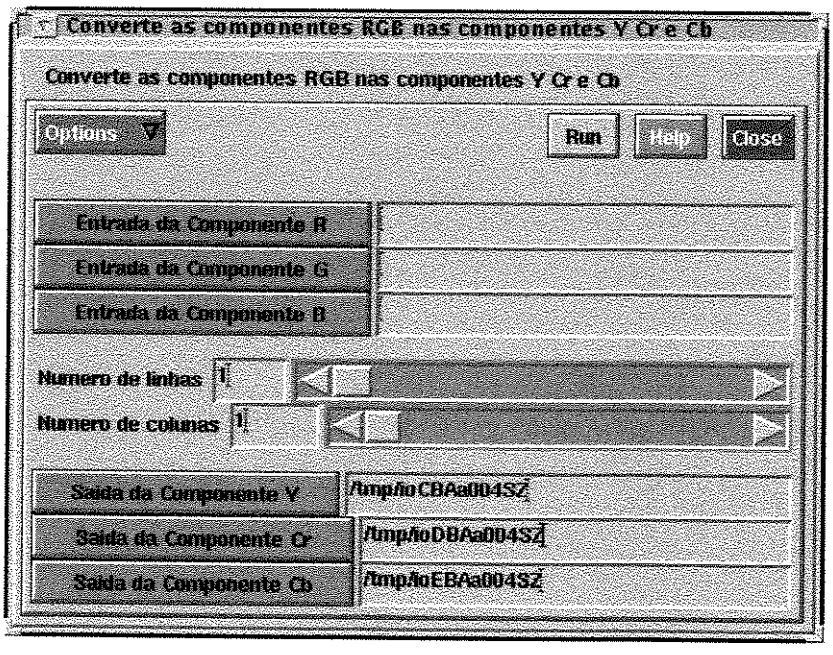

Figura A.15: Interface gráfica de usuário, pane, da kroutine rgb_ycrcb.

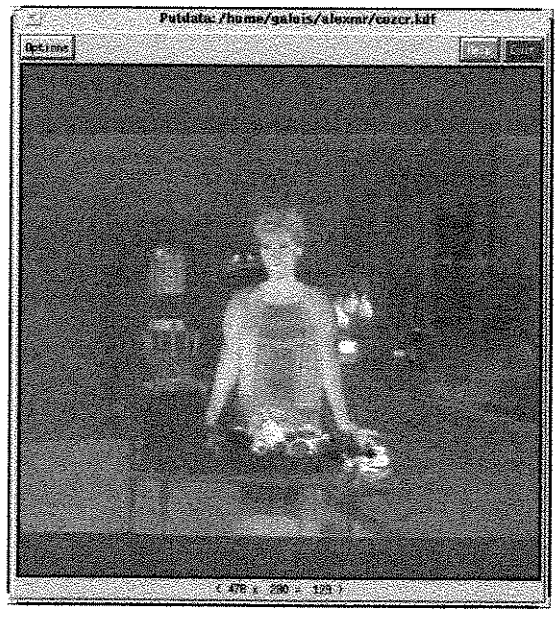

Figura A.16: Componente Cr obtida na saída do programa RGBYCrCb.

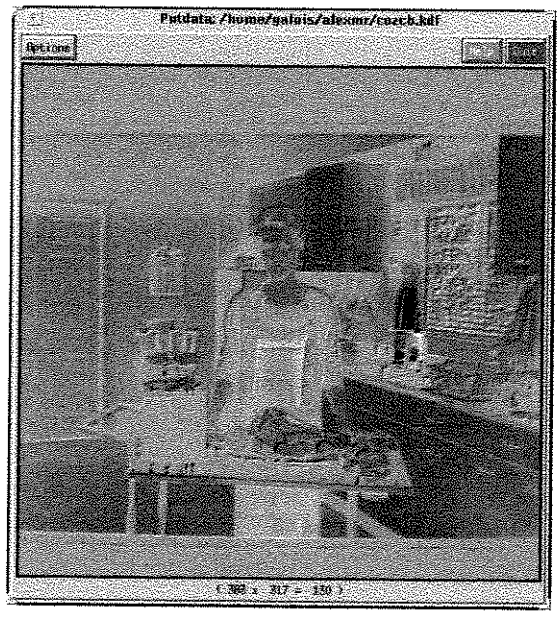

Figura A.17: Componente Cb obtida na saída do programa RGBYCrCb.

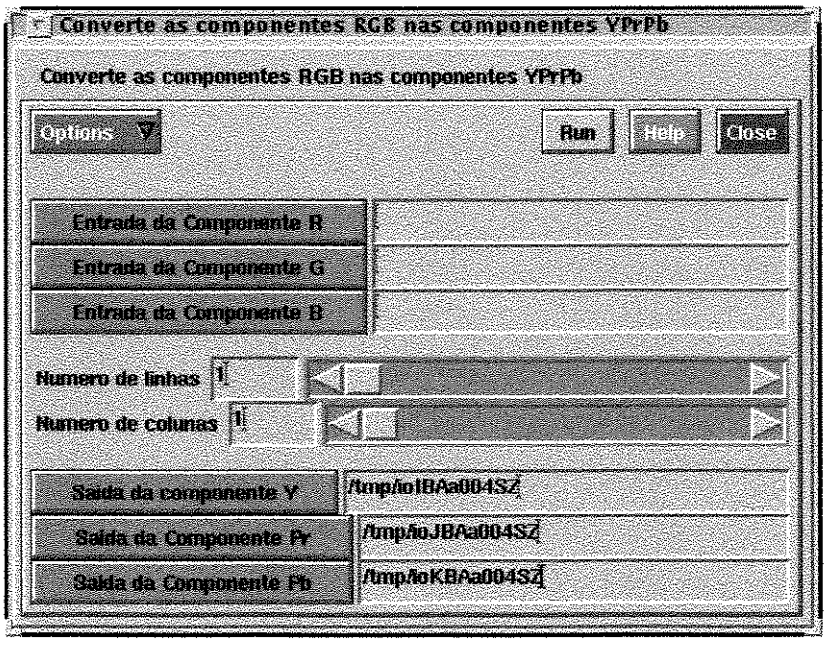

Figura A.18: Interface gráfica de usuário, pane, da kroutine rgb_yprpb.

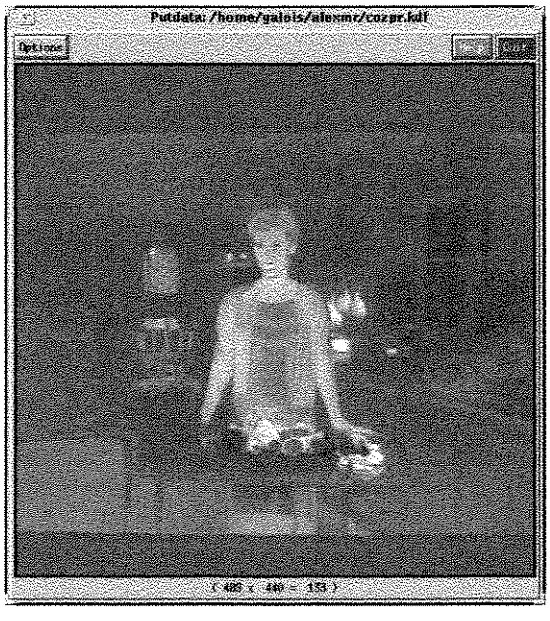

Figura A.19: Componente Pr obtida na saída do programa RGBYPrPb.

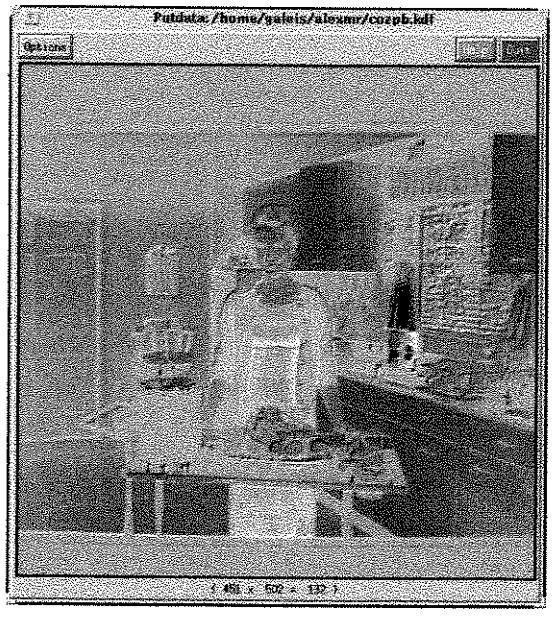

Figura A.20: Componente Pb obtida na saída do programa RGBYPrPb.

A interface gráfica de usuário do programa YUV_RGB é mostrada na Figura A.21.
A interface gráfica de usuário do programa YIQ_RGB é mostrada na Figura A.22.
A interface gráfica de usuário do programa YCrCb_RGB é mostrada na Figura A.23.
A interface gráfica de usuário do programa YPrPb_RGB é mostrada na Figura A.24.

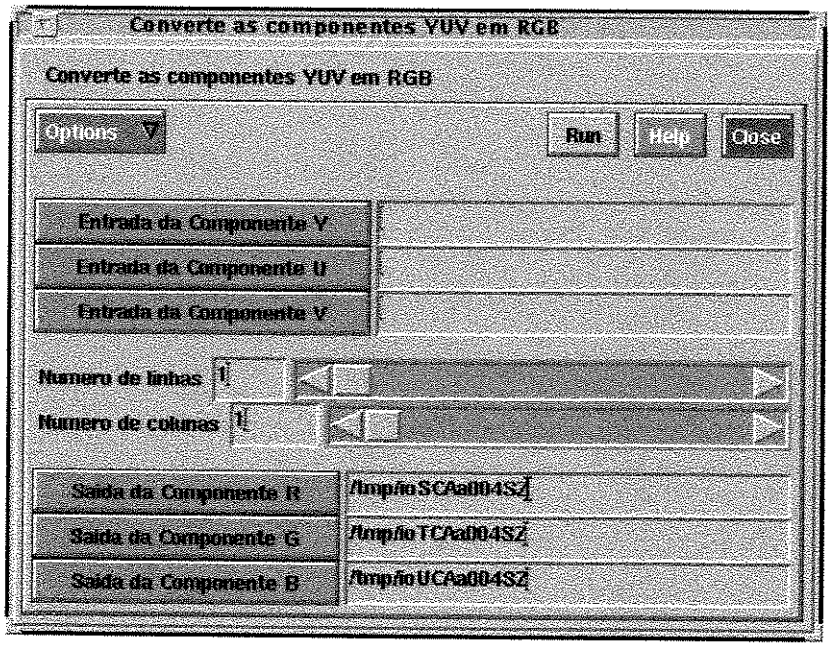

Figura A.21: Interface gráfica de usuário, pane, da kroutine yuv_rgb.

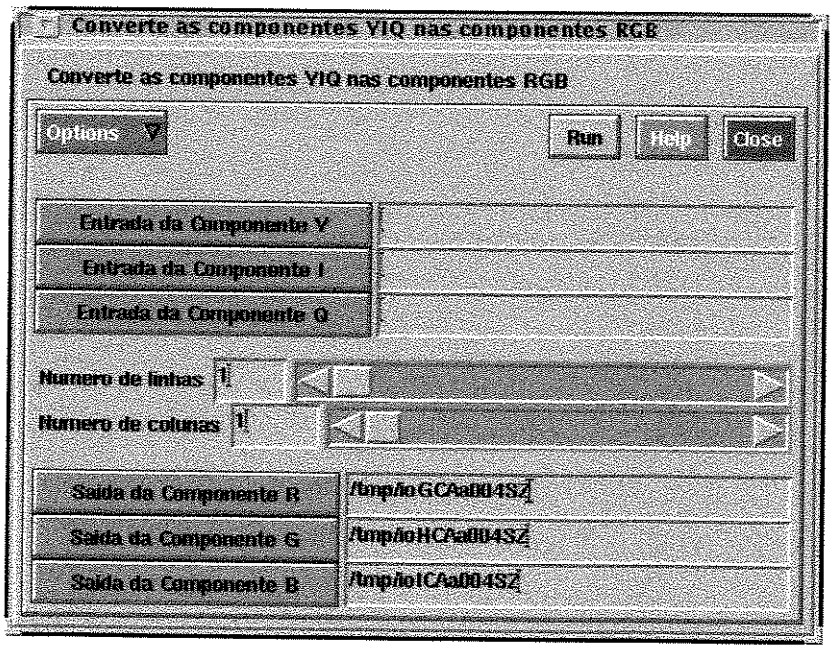

Figura A.22: Interface gráfica de usuário, pane, da kroutine yiq_rgb.

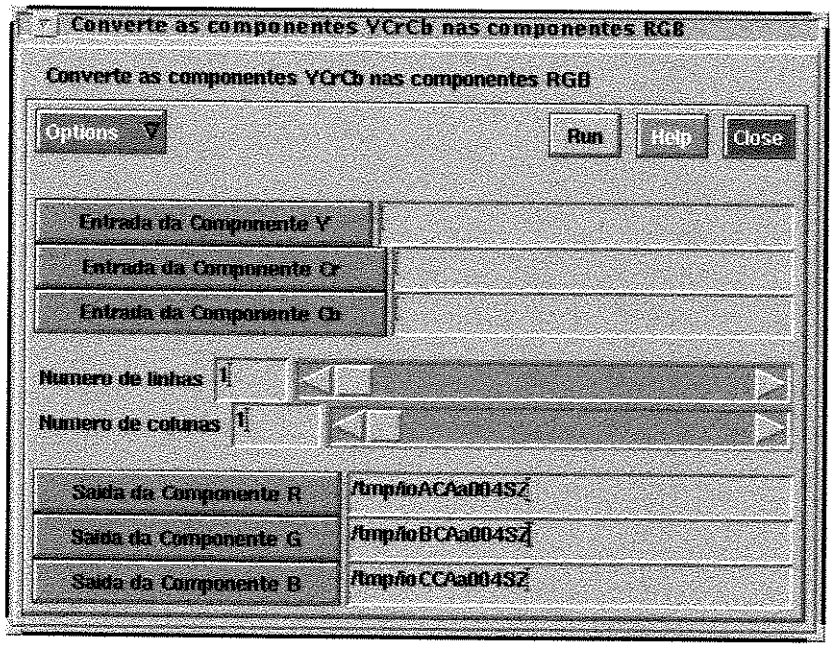

Figura A.23: Interface gráfica de usuário, pane, da kroutine ycrcb_rgb.

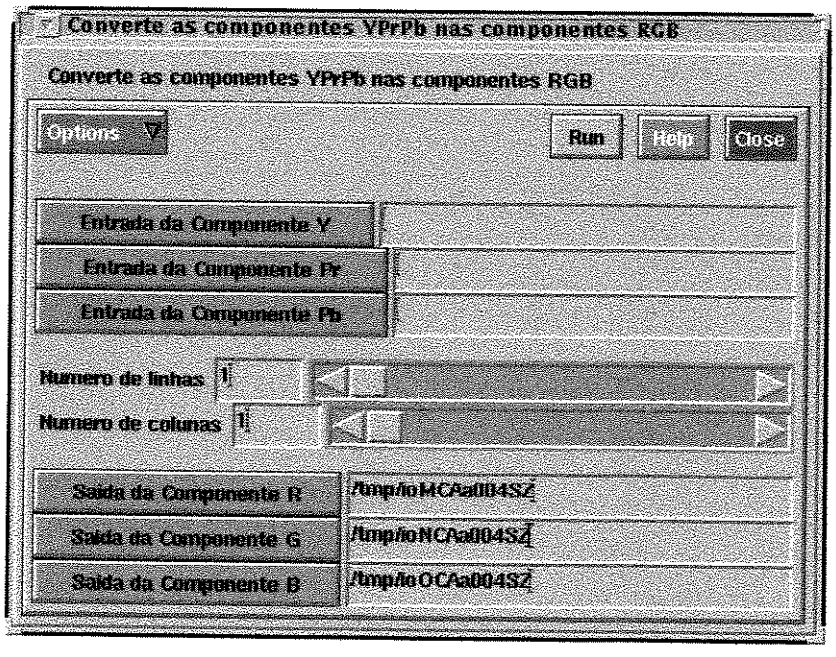

Figura A.24: Interface gráfica de usuário, pane, da kroutine yprpb_rgb.

Bibliografia

[1] Khoral Research, Inc., *http ://www.khoral.com*, 1998

[2] The Khoros Groups, *KHOROS MANUAL - User's Manual*, University of New Mexico, Albuquerque, 1991

[3] The Khoros Groups, *KHOROS MANUAL - Programmer's Manual*, University of New Mexico, Albuquerque, 1991

[4] Khoral Research, Inc., *Toolbox Programming Manual*, Albuquerque, 1994

[5] Khoral Research, Inc., *Khoros Pro User's Guide*, Albuquerque, 1996

[6] Khoral Research, Inc., *Programming Services Volume II, Data Services*, Albuquerque,1994

[7] Santos, Rafael , *Khoros Programming Tutorial*, http ://mickey08.mickey.ai.kyutech.ac.jp, 1997

[8] KRI; ISTEC; Jordán, Ramiro; Lotufo, Roberto , *DIP with khoros 2*, via ftp.khoral.com em /pub/khoros/khoros2/contrib/toolboxes/dipcourse

[9] Mizrahi, Victorine V., *Treinamento em Linguagem C - Módulo 1* , McGraw-Hill, Inc, São Paulo, 1990

[10] Mizrahi, Victorine V., *Treinamento em Linguagem C - Módulo 2* , McGraw-Hill, Inc, São Paulo, 1990

[11] Aitken, Peter; Jones,Bradley, *C- Guia do Programador*, Berkeley, Rio de Janeiro, 1994

[12] Donohoe, Gregory W., *Image Processing Short Course with Khoros*, University of New Mexico, Albuquerque,1992

[13] Jordán, Ramiro; Lotufo, Roberto , *Processamento Digital de Imagem com khoros 2*, UNICAMP, 1995

[14] Cardoso, Alexandra M. R.; Nascimento, Ayres M. A.; Iano, Yuzo; *Desenvolvimento de Programas em Linguagem C no Ambiente de Programação Visual - Khoros,* SENACITEL'98, Valdivia, Chile,1998 (aceito)

[15] Cardoso, Alexandra M. R.; Iano, Yuzo; Nascimento, Ayres M. A.; Kemper, Guillermo Vasquez, *Desarollo de programas de linguaje C y en el ambiente de programa visual - Khoros,* Centro de Informacion Tecnologica (CIT), Revista Internacial arbitrada em idioma Español, ISSN0716-8756,1998(submetido)

Programas principais

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for rgb_yuv
  >>>>
  >>>> Private:
  >>>>     main
  >>>>
  >>>>  Static:
  >>>>  Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "rgb_yuv.h"

clui_info_struct *clui_info = NULL;

/*-------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes RGB em YUV
|
|     Purpose: main program for rgb_yuv
|
|      Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente R}
|                       int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente G}
|                       int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente B}
|                       int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|
|                       int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente Y}
|                       int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente U}
|                       int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de saida da componente V}
|                       int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|      Output:
|     Returns:
|
|   Written By: Ayres
|        Date: Jun 29, 1998
| Modifications:
|
-------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "rgb_yuv",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*U,*V,*YN,*UN,*VN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b,**y,**u,**v;
   float **yn,**un,**vn;
   int i,j,n;
   int linha,coluna;
   /*unsigned char value;*/
   unsigned char *planeY=NULL;
   unsigned char *planeU=NULL;
   unsigned char *planeV=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqY,*nome_arqU,*nome_arqV;

/* Valores da matriz de conversao RGB->PAL_M */

   d00= 0.299;   d01= 0.587;   d02= 0.114;   /* NORMALIZADO */
```

```
   d10=-0.147;  d11=-0.289;  d12= 0.436;  /* NORMALIZADO */
   d20= 0.500;  d21=-0.4187; d22=-0.0813; /* NORMALIZADO */

/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(rgb_yuv_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "rgb_yuv",
                        rgb_yuv_usage_additions, rgb_yuv_get_args,
                        rgb_yuv_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */

            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj3);
            kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);
/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente R no formato UNSIGNED CHAR */
  do {
    /* nome_arq = ler_str("nome do arquivo de entrada da componente R");
     linha = ler_int("numero de linhas",1,512);
     coluna = ler_int("numero de colunas",1,1024);*/
     nome_arq = clui_info->i1_file;
     R = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!R);

  /*printf("\nMatriz R :");
  mostra_mat(R);*/

/*************************************************************************/
/* Abre e ler um arquivo da componente G no formato UNSIGNED CHAR */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente G");*/
    nome_arq = clui_info->i2_file;
    G = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!G);

  /*printf("\nMatriz G :");
  mostra_mat(G);*/

/*************************************************************************/
/* Abre e ler um arquivo da componente B no formato unsigned char */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente B");*/
    nome_arq = clui_info->i3_file;
    B = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!B);

  /*printf("\nMatriz B :");
  mostra_mat(B);*/

/*************************************************************************/
/* Aloca espaco para Y, abre arquivo, extrai a componente Y e grava
  no formato INT */
    YN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Y = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
   /* nome_arqY = ler_str("nome do arquivo de saida Y");*/
    nome_arqY = clui_info->o1_file;
```

```
    /* Extrai a componente Y a partir de R, G e B */
      COMPOE(R,G,B,YN,d00,d01,d02,R->linha,R->coluna,unsigned char,float);

      /*printf("\nMatriz YN :");
      mostra_mat(YN);*/

      yn = (float **)YN->ptr_mat;
      y = (unsigned char **)Y->ptr_mat;
      for(i = 0; i < R->linha; i++)
              for(j = 0; j < R->coluna; j++)
                y[i][j] = ROUND(yn[i][j]);

  /* Grava a matriz da componente Y */
      grava_mat(Y,nome_arqY);

      planeY = (unsigned char *)kmalloc(Y->coluna*Y->linha*sizeof(unsigned char));
      for(i = 0; i < Y->linha; i++){
               memcpy(&planeY[i*Y->coluna],Y->ptr_mat[i],sizeof(unsigned char)*Y->coluna);
      }
      kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeY);

      /*for(i = 0; i < Y->linha; i++){
              for(j = 0; j < Y->coluna; j++){
                 value = (unsigned char)Y->ptr_mat[i][j];
                 kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
              }
      }*/

      /*printf("\nMatriz Y :");
      mostra_mat(Y);*/

      libera_mat(YN);
      libera_mat(Y);
  /*********************************************************************/
  /* Aloca espaco para U, abre arquivo, extrai a componante U e grava
    no formato INT */
      UN = aloca_mat(R->linha,R->coluna,sizeof(float));
      U = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
      /*nome_arqU = ler_str("nome do arquivo de saida U");*/
      nome_arqU = clui_info->o2_file;
    /* Extrai a componente U a partir de R, G e B */
      COMPOE(R,G,B,UN,d10,d11,d12,R->linha,R->coluna,unsigned char,float);

      /* printf("\nMatriz UN :");
      mostra_mat(UN);*/

      ESCALAR_MAT(UN,UN,+,127.5,R->linha,R->coluna,0,0,float,float);
      un = (float **)UN->ptr_mat;
      u = (unsigned char **)U->ptr_mat;

      for(i = 0; i < R->linha; i++)
              for(j = 0; j < R->coluna; j++)
                u[i][j] =  ROUND(un[i][j]);
  /* Grava a matriz da componente U */
      grava_mat(U,nome_arqU);

      planeU = (unsigned char *)kmalloc(U->coluna*U->linha*sizeof(unsigned char));
      for(i = 0; i < U->linha; i++){
               memcpy(&planeU[i*U->coluna],U->ptr_mat[i],sizeof(unsigned char)*U->coluna);
      }
      kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeU);

      /*for(i = 0; i < U->linha; i++){
              for(j = 0; j < U->coluna; j++){
                 value = (unsigned char)U->ptr_mat[i][j];
                 kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
              }
      }*/
      /*printf("\nMatriz U :");
      mostra_mat(U);*/

      libera_mat(UN);
      libera_mat(U);
  /*********************************************************************/
  /* Aloca espaco para V, abre arquivo, extrai a componante V e grava
    no formato INT */
      VN = aloca_mat(R->linha,R->coluna,sizeof(float));
      V = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
      /*nome_arqV = ler_str("nome do arquivo de saida V");*/
      nome_arqV = clui_info->o3_file;
    /* Extrai a componente V a partir de R, G e B */
      COMPOE(R,G,B,VN,d20,d21,d22,R->linha,R->coluna,unsigned char,float);

      /* printf("\nMatriz VN :");
      mostra_mat(VN);*/

      ESCALAR_MAT(VN,VN,+,127.5,R->linha,R->coluna,0,0,float,float);
      vn = (float **)VN->ptr_mat;
      v = (unsigned char **)V->ptr_mat;
      for(i = 0; i < R->linha; i++)
              for(j = 0; j < R->coluna; j++)
                v[i][j] =  ROUND(vn[i][j]);
  /* Grava a matriz da componente V */
      grava_mat(V,nome_arqV);

      planeV = (unsigned char *)kmalloc(V->coluna*V->linha*sizeof(unsigned char));
      for(i = 0; i < V->linha; i++){
              memcpy(&planeV[i*V->coluna],V->ptr_mat[i],sizeof(unsigned char)*V->coluna);
      }
```

```
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeV);

    /*for(i = 0; i < V->linha; i++){
            for(j = 0; j < V->coluna; j++){
               value = (unsigned char)V->ptr_mat[i][j];
               kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
    /* printf("\nMatriz V :");
     mostra_mat(V);*/

    libera_mat(VN);
    libera_mat(V);
/* -main_library_call_end */

/* -main_after_lib_call */
    libera_mat(R);
    libera_mat(G);
    libera_mat(B);
    /* Step 7: fecha o arquivo out_obj1  */
    if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
    kpds_close_object(out_obj1);

    /* Step 8: fecha o arquivo out_obj2 */
    if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
    kpds_close_object(out_obj2);

    /* Step 9: fecha o arquivo out_obj3 */
    if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
    kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}
```

```
/*-------------------------------------------------------------
|
| Routine Name: rgb_yuv_usage_additions
|
|     Purpose: Prints usage additions in rgb_yuv_usage routine
|
|      Input: None
|
|     Output: None
|  Written By: ghostwriter -oname rgb_yuv
|       Date: Jun 29, 1998
| Modifications:
|
-------------------------------------------------------------*/
void rgb_yuv_usage_additions(void)
{
        kfprintf(kstderr, "\tConverte as componentes RGB em YUV\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-------------------------------------------------------------
|
| Routine Name: rgb_yuv_free_args
|
|     Purpose: Frees CLUI struct allocated in rgb_yuv_get_args()
|
|      Input: None
|
|     Output: None
|  Written By: ghostwriter -oname rgb_yuv
|       Date: Jun 29, 1998
| Modifications:
|
-------------------------------------------------------------*/
/* ARGSUSED */
void
rgb_yuv_free_args(
  kexit_status status,
  kaddr       client_data)
{

        /* do the wild and free thing */
        if (clui_info != NULL)
        {
                kfree_and_NULL(clui_info->i1_file);
                kfree_and_NULL(clui_info->i2_file);
                kfree_and_NULL(clui_info->i3_file);
```

```
                kfree_and_NULL(clui_info->o1_file);
                kfree_and_NULL(clui_info->o2_file);
                kfree_and_NULL(clui_info->o3_file);
                kfree_and_NULL(clui_info);
        }

/* -free_handler_additions */
/* -free_handler_additions_end */
}



1
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for rgb_yuv
  >>>>
  >>>>   Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>        Date: Jun 29, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _rgb_yuv_h_
#define _rgb_yuv_h_

    /*---------------------------------------*
    |     #includes
    ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "maniparq.h"
/* -include_includes_end */


    /*---------------------------------------*
    |     #defines
    ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/rgb_yuv/uis/rgb_yuv.pane"

    /*---------------------------------------*
    |     typedefs
    ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 *  Arquivo de entrada da componente R (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente R FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente R FLAG */

/*
 *  Arquivo de entrada da componente G (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente G FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente G FLAG */

/*
 *  Arquivo de entrada da componente B (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente B FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente B FLAG */

/*
 *  Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 *  Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 *  Arquivo de saida da componente Y (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente Y FILENAME */
int          o1_flag;   /* Arquivo de saida da componente Y FLAG */

/*
 *  Arquivo de saida da componente U (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente U FILENAME */
int          o2_flag;   /* Arquivo de saida da componente U FLAG */
```

```
/*
 *  Arquivo de saida da componente V (required outfile)
 */
char        *o3_file;   /* Arquivo de saida da componente V FILENAME */
int          o3_flag;   /* Arquivo de saida da componente V FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


      /*----------------------------------------*
      |     global variable declarations
      ----------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


      /*----------------------------------------*
      |     macros
      ----------------------------------------*/

/* -include_macros */
/* -include_macros_end */

      /*----------------------------------------*
      |     routine definitions
      ----------------------------------------*/

void main PROTO((int, char **));
void rgb_yuv_get_args PROTO((kform *));
void rgb_yuv_usage_additions PROTO((void));
void rgb_yuv_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for rgb_yiq
  >>>>
  >>>> Private:
  >>>>     main
  >>>>
  >>>> Static:
  >>>> Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "rgb_yiq.h"

clui_info_struct *clui_info = NULL;

/*-------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes RGB nas componentes YIQ
|
|    Purpose: main program for rgb_yiq
|
|     Input:
|                     char *clui_info->i1_file; {Arquivo de entrada da componente R}
|                     int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                     char *clui_info->i2_file; {Arquivo de entrada da componente G}
|                     int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                     char *clui_info->i3_file; {Arquivo de entrada da componente B}
|                     int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                     int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                     int clui_info->int1_flag; {TRUE if -int1 specified}
|
|                     int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                     int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                     char *clui_info->o1_file; {Arquivo de saida da componente Y}
|                     int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                     char *clui_info->o2_file; {Arquivo de saida da componente I}
|                     int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                     char *clui_info->o3_file; {Arquivo de saida da componente Q}
|                     int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|    Output:
|   Returns:
|
|  Written By:
|      Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

        kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "rgb_yuv",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*I,*Q,*YN,*IN,*QN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b,**yy,**ii,**qq;
   float **yn,**in,**qn;
   int i,j,n;
   int linha,coluna;
   /*unsigned char value;*/
   unsigned char *planeY=NULL;
   unsigned char *planeI=NULL;
   unsigned char *planeQ=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqY,*nome_arqI,*nome_arqQ;

/* Valores da matriz de conversao RGB->NTSC */

   d00=0.299;   d01= 0.587;   d02= 0.114;   /* NORMALIZADO */
```

```
   d10=0.50;   d11=-0.2307; d12=-0.2693; /* NORMALIZADO */
   d20=0.2027; d21=-0.50;   d22= 0.2973; /* NORMALIZADO */
/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(rgb_yiq_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "rgb_yiq",
                        rgb_yiq_usage_additions, rgb_yiq_get_args,
                        rgb_yiq_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
            kpds_create_value(out_obj3);
                  kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
                  kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);

/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente R no formato UNSIGNED CHAR */
  do {
    /* nome_arq = ler_str("nome do arquivo de entrada da componente R");
     linha = ler_int("numero de linhas",1,512);
     coluna = ler_int("numero de colunas",1,1024);*/
     nome_arq = clui_info->i1_file;
     R = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!R);
/*
  printf("\nMatriz R :");
  mostra_mat(R);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente G no formato UNSIGNED CHAR */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente G");*/
     nome_arq = clui_info->i2_file;
     G = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!G);
/*
  printf("\nMatriz G :");
  mostra_mat(G);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente B no formato UNSIGNED CHAR */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente B");*/
     nome_arq = clui_info->i3_file;
     B = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!B);
/*
  printf("\nMatriz B :");
  mostra_mat(B);
*/
/*************************************************************************/
/*************************************************************************/
/* Aloca espaco para Y, abre arquivo, extrai a componante Y e grava
  no formato int */
     YN = aloca_mat(R->linha,R->coluna,sizeof(float));
     Y = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
     /*nome_arqY = ler_str("nome do arquivo de saida Y");*/
     nome_arqY = clui_info->o1_file;
```

```
  /* Extari a componente Y a partir de R, G e B */
    COMPOE(R,G,B,YN,d00,d01,d02,R->linha,R->coluna,unsigned char,float);

    /* printf("\nMatriz YN :");
    mostra_mat(YN);*/

    yn = (float **)YN->ptr_mat;
    yy = (unsigned char **)Y->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              yy[i][j] = ROUND(yn[i][j]);
/* Grava a matriz da componente Y */
    grava_mat(Y,nome_arqY);

    planeY = (unsigned char *)kmalloc(Y->coluna*Y->linha*sizeof(unsigned char));
    for(i = 0; i < Y->linha; i++){
            memcpy(&planeY[i*Y->coluna],Y->ptr_mat[i],sizeof(unsigned char)*Y->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeY);

    /*for(i = 0; i < Y->linha; i++){
            for(j = 0; j < Y->coluna; j++){
              value = (unsigned char)Y->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

   /* printf("\nMatriz Y :");
    mostra_mat(Y);*/

    libera_mat(Y);
    libera_mat(YN);
/*********************************************************************/
/* Aloca espaco para I, abre arquivo, extrai a componante I e grava
  no formato int */
    IN = aloca_mat(R->linha,R->coluna,sizeof(float));
    I = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
    /*nome_arqI = ler_str("nome do arquivo de saida I");*/
    nome_arqI = clui_info->o2_file;
  /* Extari a componente I a partir de R, G e B */
    COMPOE(R,G,B,IN,d10,d11,d12,R->linha,R->coluna,unsigned char,float);
    ESCALAR_MAT(IN,IN,+,127.5,R->linha,R->coluna,0,0,float,float);

    /*printf("\nMatriz IN :");
    mostra_mat(IN);*/

    in = (float **)IN->ptr_mat;
    ii = (unsigned char **)I->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              ii[i][j] = ROUND(in[i][j]);
/* Grava a matriz da componente I */
    grava_mat(I,nome_arqI);

    planeI = (unsigned char *)kmalloc(I->coluna*I->linha*sizeof(unsigned char));
    for(i = 0; i < I->linha; i++){
            memcpy(&planeI[i*I->coluna],I->ptr_mat[i],sizeof(unsigned char)*I->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeI);

    /*for(i = 0; i < I->linha; i++){
            for(j = 0; j < I->coluna; j++){
              value = (unsigned char)I->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

   /* printf("\nMatriz I :");
    mostra_mat(I);*/

    libera_mat(I);
    libera_mat(IN);
/*********************************************************************/
/* Aloca espaco para Q, abre arquivo, extrai a componante Q e grava
  no formato int */
    QN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Q = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
    /*nome_arqQ = ler_str("nome do arquivo de saida Q");*/
    nome_arqQ = clui_info->o3_file;
  /* Extari a componente Q a partir de R, G e B */
    COMPOE(R,G,B,QN,d20,d21,d22,R->linha,R->coluna,unsigned char,float);
    ESCALAR_MAT(QN,QN,+,127.5,R->linha,R->coluna,0,0,float,float);

    /*printf("\nMatriz QN :");
    mostra_mat(QN);*/

    qn = (float **)QN->ptr_mat;
    qq = (unsigned char **)Q->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              qq[i][j] = ROUND(qn[i][j]);
/* Grava a matriz da componente Q */
    grava_mat(Q,nome_arqQ);

    planeQ = (unsigned char *)kmalloc(Q->coluna*Q->linha*sizeof(unsigned char));
    for(i = 0; i < Q->linha; i++){
            memcpy(&planeQ[i*Q->coluna],Q->ptr_mat[i],sizeof(unsigned char)*Q->coluna);
    }
```

```
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeQ);

    /*for(i = 0; i < Q->linha; i++){
            for(j = 0; j < Q->coluna; j++){
               value = (unsigned char)Q->ptr_mat[i][j];
               kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

    /* printf("\nMatriz Q :");
     mostra_mat(Q);*/

    libera_mat(Q);
    libera_mat(QN);
/* -main_library_call_end */

/* -main_after_lib_call */
   libera_mat(R);
   libera_mat(G);
   libera_mat(B);

   /* Step 7: fecha o arquivo out_obj1  */
   if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj1);

   /* Step 8: fecha o arquivo out_obj2 */
   if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj2);

   /* Step 9: fecha o arquivo out_obj3 */
   if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}
```

```
/*-----------------------------------------------------------
|
| Routine Name: rgb_yiq_usage_additions
|
|     Purpose: Prints usage additions in rgb_yiq_usage routine
|
|      Input: None
|
|     Output: None
|  Written By: ghostwriter -oname rgb_yiq
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------*/
void rgb_yiq_usage_additions(void)
{
        kfprintf(kstderr, "\tConverte as componentes RGB nas componentes YIQ\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-----------------------------------------------------------
|
| Routine Name: rgb_yiq_free_args
|
|     Purpose: Frees CLUI struct allocated in rgb_yiq_get_args()
|
|      Input: None
|
|     Output: None
|  Written By: ghostwriter -oname rgb_yiq
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------*/
/* ARGSUSED */
void
rgb_yiq_free_args(
  kexit_status status,
  kaddr       client_data)
{

        /* do the wild and free thing */
        if (clui_info != NULL)
        {
                kfree_and_NULL(clui_info->il_file);
```

```
                kfree_and_NULL(clui_info->i2_file);
                kfree_and_NULL(clui_info->i3_file);
                kfree_and_NULL(clui_info->o1_file);
                kfree_and_NULL(clui_info->o2_file);
                kfree_and_NULL(clui_info->o3_file);
                kfree_and_NULL(clui_info);
        }

/* -free_handler_additions */
/* -free_handler_additions_end */
}
```

1

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for rgb_yiq
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>         Date: Jul 16, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _rgb_yiq_h_
#define _rgb_yiq_h_

    /*---------------------------------------*
     |     #includes
     ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/library/coreslib/src/maniparq.h"
/* -include_includes_end */


    /*---------------------------------------*
     |     #defines
     ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/rgb_yiq/uis/rgb_yiq.pane"
    /*---------------------------------------*
     |     typedefs
     ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente R (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente R FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente R FLAG */

/*
 * Arquivo de entrada da componente G (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente G FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente G FLAG */

/*
 * Arquivo de entrada da componente B (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente B FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente B FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente Y (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente Y FILENAME */
int          o1_flag;   /* Arquivo de saida da componente Y FLAG */

/*
 * Arquivo de saida da componente I (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente I FILENAME */
int          o2_flag;   /* Arquivo de saida da componente I FLAG */

/*
 * Arquivo de saida da componente Q (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de saida da componente Q FILENAME */
int          o3_flag;   /* Arquivo de saida da componente Q FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


      /*----------------------------------------*
      |     global variable declarations
      ----------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


      /*----------------------------------------*
      |     macros
      ----------------------------------------*/

/* -include_macros */
/* -include_macros_end */

      /*----------------------------------------*
      |     routine definitions
      ----------------------------------------*/

void main PROTO((int, char **));
void rgb_yiq_get_args PROTO((kform *));
void rgb_yiq_usage_additions PROTO((void));
void rgb_yiq_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

1

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for rgbycrcb
  >>>>
  >>>> Private:
  >>>>     main
  >>>>
  >>>> Static:
  >>>> Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "rgbycrcb.h"

clui_info_struct *clui_info = NULL;

/*-------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes RGB nas componentes Y Cr e Cb
|
|     Purpose: main program for rgbycrcb
|
|      Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente R}
|                       int  clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente G}
|                       int  clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente B}
|                       int  clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|
|                       int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente Y}
|                       int  clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente Cr}
|                       int  clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de saida da componente Cb}
|                       int  clui_info->o3_flag; {TRUE if -o3 specified}
|
|     Output:
|    Returns:
|
|  Written By:
|       Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "rgb_yuv",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*Cr,*Cb,*YN,*CrN,*CbN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b,**yy,**cr,**cb;
   float **yn,**crn,**cbn;
   int i,j,n;
   int linha,coluna;
   /*unsigned char value;*/
   unsigned char *planeY=NULL;
   unsigned char *planeCr=NULL;
   unsigned char *planeCb=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqY,*nome_arqCr,*nome_arqCb;

/* Valores da matriz de conversao RGB->YCrCb */

   d00= 0.25831;   d01= 0.50321;   d02= 0.097289;  /* NORMALIZADO */
```

```
   d10= 0.43947;  d11=-0.36902;  d12=-0.07045;  /* NORMALIZADO */
   d20=-0.14761;  d21=-0.29187;  d22= 0.43948;  /* NORMALIZADO */
/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(rgbycrcb_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "rgbycrcb",
                        rgbycrcb_usage_additions, rgbycrcb_get_args,
                        rgbycrcb_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj3);
            kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);
/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente R no formato UNSIGNED CHAR */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente R");
    linha = ler_int("numero de linhas",1,512);
    coluna = ler_int("numero de colunas",1,1024);*/
    nome_arq = clui_info->i1_file;
    R = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!R);
/*
  printf("\nMatriz R :");
  mostra_mat(R);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente G no formato UNSIGNED CHAR */
  do {
   /* nome_arq = ler_str("nome do arquivo de entrada da componente G");*/
    nome_arq = clui_info->i2_file;
    G = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!G);
/*
  printf("\nMatriz G :");
  mostra_mat(G);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente B no formato UNSIGNED CHAR */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente B");*/
    nome_arq = clui_info->i3_file;
    B = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!B);
/*
  printf("\nMatriz B :");
  mostra_mat(B);
*/
/*************************************************************************/
/*************************************************************************/
/* Aloca espaco para Y, abre arquivo, extrai a componante Y e grava
  no formato int */
    YN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Y = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
    /*nome_arqY = ler_str("nome do arquivo de saida Y");*/
    nome_arqY = clui_info->o1_file;
  /* Extari a componente Y a partir de R, G e B */
```

```
    COMPOE(R,G,B,YN,d00,d01,d02,R->linha,R->coluna,unsigned char,float);
    ESCALAR_MAT(YN,YN,+,16.0,R->linha,R->coluna,0,0,float,float);
    yn = (float **)YN->ptr_mat;
    yy = (unsigned char **)Y->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              yy[i][j] = ROUND(yn[i][j]);
/* Grava a matriz da componente Y */
    grava_mat(Y,nome_arqY);

    planeY = (unsigned char *)kmalloc(Y->coluna*Y->linha*sizeof(unsigned char));
    for(i = 0; i < Y->linha; i++){
            memcpy(&planeY[i*Y->coluna],Y->ptr_mat[i],sizeof(unsigned char)*Y->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeY);

    /*for(i = 0; i < Y->linha; i++){
            for(j = 0; j < Y->coluna; j++){
              value = (unsigned char)Y->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz Y :");
    mostra_mat(Y);
*/
    libera_mat(Y);
    libera_mat(YN);
/*********************************************************************/
/* Aloca espaco para Cr, abre arquivo, extrai a componante Cr e grava
  no formato unsigned char */
    CrN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Cr = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
   /* nome_arqCr = ler_str("nome do arquivo de saida Cr");*/
    nome_arqCr = clui_info->o2_file;
  /* Extari a componente Cr a partir de R, G e B */
    COMPOE(R,G,B,CrN,d10,d11,d12,R->linha,R->coluna,unsigned char,float);
    ESCALAR_MAT(CrN,CrN,+,128.0,R->linha,R->coluna,0,0,float,float);
    crn = (float **)CrN->ptr_mat;
    cr = (unsigned char **)Cr->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              cr[i][j] = ROUND(crn[i][j]);
/* Grava a matriz da componente Cr */
    grava_mat(Cr,nome_arqCr);

    planeCr = (unsigned char *)kmalloc(Cr->coluna*Cr->linha*sizeof(unsigned char));
    for(i = 0; i < Cr->linha; i++){
            memcpy(&planeCr[i*Cr->coluna],Cr->ptr_mat[i],sizeof(unsigned char)*Cr->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeCr);

    /*for(i = 0; i < Cr->linha; i++){
            for(j = 0; j < Cr->coluna; j++){
              value = (unsigned char)Cr->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz Cr :");
    mostra_mat(Cr);
*/
    libera_mat(Cr);
    libera_mat(CrN);
/*********************************************************************/
/* Aloca espaco para Cb, abre arquivo, extrai a componante Cb e grava
  no formato unsigned char */
    CbN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Cb = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
   /* nome_arqCb = ler_str("nome do arquivo de saida Cb");*/
    nome_arqCb = clui_info->o3_file;
  /* Extari a componente Cb a partir de R, G e B */
    COMPOE(R,G,B,CbN,d20,d21,d22,R->linha,R->coluna,unsigned char,float);
    ESCALAR_MAT(CbN,CbN,+,128.0,R->linha,R->coluna,0,0,float,float);
    cbn = (float **)CbN->ptr_mat;
    cb = (unsigned char **)Cb->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              cb[i][j] = ROUND(cbn[i][j]);
/* Grava a matriz da componente Cb */
    grava_mat(Cb,nome_arqCb);

    planeCb = (unsigned char *)kmalloc(Cb->coluna*Cb->linha*sizeof(unsigned char));
    for(i = 0; i < Cb->linha; i++){
            memcpy(&planeCb[i*Cb->coluna],Cb->ptr_mat[i],sizeof(unsigned char)*Cb->coluna);
    }
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeCb);

    /*for(i = 0; i < Cb->linha; i++){
            for(j = 0; j < Cb->coluna; j++){
              value = (unsigned char)Cb->ptr_mat[i][j];
              kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz Cb :");
    mostra_mat(Cb);
*/
```

```
    libera_mat(Cb);
    libera_mat(CbN);

/* -main_library_call_end */

/* -main_after_lib_call */
   libera_mat(R);
   libera_mat(G);
   libera_mat(B);
   /* Step 7: fecha o arquivo out_obj1  */
   if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj1);

   /* Step 8: fecha o arquivo out_obj2 */
   if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj2);

   /* Step 9: fecha o arquivo out_obj3 */
   if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}


/*-----------------------------------------------------------
|
| Routine Name: rgbycrcb_usage_additions
|
|    Purpose: Prints usage additions in rgbycrcb_usage routine
|
|     Input: None
|
|    Output: None
|  Written By: ghostwriter -oname rgbycrcb
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------*/
void rgbycrcb_usage_additions(void)
{
        kfprintf(kstderr, "\tConverte as componentes RGB nas componentes Y Cr e Cb\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-----------------------------------------------------------
|
| Routine Name: rgbycrcb_free_args
|
|    Purpose: Frees CLUI struct allocated in rgbycrcb_get_args()
|
|     Input: None
|
|    Output: None
|  Written By: ghostwriter -oname rgbycrcb
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------*/
/* ARGSUSED */
void
rgbycrcb_free_args(
   kexit_status status,
   kaddr       client_data)
{

        /* do the wild and free thing */
        if (clui_info != NULL)
        {
                kfree_and_NULL(clui_info->i1_file);
                kfree_and_NULL(clui_info->i2_file);
                kfree_and_NULL(clui_info->i3_file);
                kfree_and_NULL(clui_info->o1_file);
                kfree_and_NULL(clui_info->o2_file);
                kfree_and_NULL(clui_info->o3_file);
                kfree_and_NULL(clui_info);
        }

/* -free_handler_additions */
/* -free_handler_additions_end */
}
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for rgbycrcb
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>        Date: Jul 16, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _rgbycrcb_h_
#define _rgbycrcb_h_

    /*---------------------------------------*
    |     #includes
    ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/library/coreslib/src/maniparq.h"
/* -include_includes_end */


    /*---------------------------------------*
    |     #defines
    ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/rgbycrcb/uis/rgbycrcb.pane"
    /*---------------------------------------*
    |     typedefs
    ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente R (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente R FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente R FLAG */

/*
 * Arquivo de entrada da componente G (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente G FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente G FLAG */

/*
 * Arquivo de entrada da componente B (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente B FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente B FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente Y (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente Y FILENAME */
int          o1_flag;   /* Arquivo de saida da componente Y FLAG */

/*
 * Arquivo de saida da componente Cr (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente Cr FILENAME */
int          o2_flag;   /* Arquivo de saida da componente Cr FLAG */

/*
 * Arquivo de saida da componente Cb (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de saida da componente Cb FILENAME */
int          o3_flag;   /* Arquivo de saida da componente Cb FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


    /*----------------------------------------*
    |    global variable declarations
    ----------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


    /*----------------------------------------*
    |    macros
    ----------------------------------------*/

/* -include_macros */
/* -include_macros_end */

    /*----------------------------------------*
    |    routine definitions
    ----------------------------------------*/

void main PROTO((int, char **));
void rgbycrcb_get_args PROTO((kform *));
void rgbycrcb_usage_additions PROTO((void));
void rgbycrcb_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for rgbyprpb
  >>>>
  >>>> Private:
  >>>>     main
  >>>>
  >>>> Static:
  >>>> Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "rgbyprpb.h"

clui_info_struct *clui_info = NULL;

/*-------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes RGB nas componentes YPrPb
|
|     Purpose: main program for rgbyprpb
|
|       Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente R}
|                       int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente G}
|                       int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente B}
|                       int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|
|                       int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente Y}
|                       int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente Pb}
|                       int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de saida da componente Pb}
|                       int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|      Output:
|     Returns:
|
|   Written By:
|        Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
/* -main_variable_list */
   char *lib = "rgb_yuv",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*Pr,*Pb,*YN,*PrN,*PbN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b,**yy,**pr,**pb;
   float **yn,**prn,**pbn;
   int i,j,n;
   int linha,coluna;
   /*unsigned char value;*/
   unsigned char *planeY=NULL;
   unsigned char *planePr=NULL;
   unsigned char *planePb=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqY,*nome_arqPr,*nome_arqPb;

/* Valores da matriz de conversao HDTV RGB->NTSC */
```

```
   d00= 0.212;   d01= 0.701;   d02= 0.087;   /* NORMALIZADO */
   d10= 0.500;   d11=-0.4448;  d12=-0.0552;  /* NORMALIZADO */
   d20=-0.1161;  d21=-0.3839;  d22= 0.500;   /* NORMALIZADO */
/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(rgbyprpb_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "rgbyprpb",
                        rgbyprpb_usage_additions, rgbyprpb_get_args,
                        rgbyprpb_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj3);
            kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);

/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente R no formato UNSIGNED CHAR */
  do {
    /* nome_arq = ler_str("nome do arquivo de entrada da componente R");
     linha = ler_int("numero de linhas",1,1250);
     coluna = ler_int("numero de colunas",1,2200);*/
     nome_arq = clui_info->i1_file;
     R = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!R);
/*
  printf("\nMatriz R :");
  mostra_mat(R);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente G no formato UNSIGNED CHAR */
  do {
    /* nome_arq = ler_str("nome do arquivo de entrada da componente G");*/
     nome_arq = clui_info->i2_file;
     G = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!G);
/*
  printf("\nMatriz G :");
  mostra_mat(G);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente B no formato UNSIGNED CHAR */
  do {
    /* nome_arq = ler_str("nome do arquivo de entrada da componente B");*/
     nome_arq = clui_info->i3_file;
     B = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!B);
/*
  printf("\nMatriz B :");
  mostra_mat(B);
*/
/*************************************************************************/
/*************************************************************************/
/* Aloca espaco para Y, abre arquivo, extrai a componante Y e grava
  no formato int */
    YN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Y = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
   /* nome_arqY = ler_str("nome do arquivo de saida Y");*/
```

```
    nome_arqY = clui_info->o1_file;
  /* Extari a componente Y a partir de R, G e B */
    COMPOE(R,G,B,YN,d00,d01,d02,R->linha,R->coluna,unsigned char,float);

    /* printf("\nMatriz YN :");
    mostra_mat(YN);*/

    yn = (float **)YN->ptr_mat;
    yy = (unsigned char **)Y->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              yy[i][j] = ROUND(yn[i][j]);
/* Grava a matriz da componente Y */
    grava_mat(Y,nome_arqY);

    planeY = (unsigned char *)kmalloc(Y->coluna*Y->linha*sizeof(unsigned char));
    for(i = 0; i < Y->linha; i++){
            memcpy(&planeY[i*Y->coluna],Y->ptr_mat[i],sizeof(unsigned char)*Y->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeY);

    /*for(i = 0; i < Y->linha; i++){
            for(j = 0; j < Y->coluna; j++){
              value = (unsigned char)Y->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

   /* printf("\nMatriz Y :");
    mostra_mat(Y);*/

    libera_mat(Y);
    libera_mat(YN);
/*********************************************************************/
/* Aloca espaco para Pr, abre arquivo, extrai a componante Pr e grava
  no formato int */
    PrN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Pr = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
    /*nome_arqPr = ler_str("nome do arquivo de saida Pr");*/
    nome_arqPr = clui_info->o2_file;
  /* Extari a componente Pr a partir de R, G e B */
    COMPOE(R,G,B,PrN,d10,d11,d12,R->linha,R->coluna,unsigned char,float);

    /*printf("\nMatriz PrN :");
    mostra_mat(PrN);*/

    ESCALAR_MAT(PrN,PrN,+,127.5,R->linha,R->coluna,0,0,float,float);

   /*  printf("\nMatriz PrN :");
    mostra_mat(PrN);*/

    prn = (float **)PrN->ptr_mat;
    pr = (unsigned char **)Pr->ptr_mat;
    for(i = 0; i < R->linha; i++)
            for(j = 0; j < R->coluna; j++)
              pr[i][j] = ROUND(prn[i][j]);
/* Grava a matriz da componente Pr */
    grava_mat(Pr,nome_arqPr);

    planePr = (unsigned char *)kmalloc(Pr->coluna*Pr->linha*sizeof(unsigned char));
    for(i = 0; i < Pr->linha; i++){
            memcpy(&planePr[i*Pr->coluna],Pr->ptr_mat[i],sizeof(unsigned char)*Pr->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planePr);

    /*for(i = 0; i < Pr->linha; i++){
            for(j = 0; j < Pr->coluna; j++){
              value = (unsigned char)Pr->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

    /*printf("\nMatriz Pr :");
    mostra_mat(Pr);*/

    libera_mat(Pr);
    libera_mat(PrN);
/*********************************************************************/
/* Aloca espaco para Pb, abre arquivo, extrai a componante Pb e grava
  no formato int */
    PbN = aloca_mat(R->linha,R->coluna,sizeof(float));
    Pb = aloca_mat(R->linha,R->coluna,sizeof(unsigned char));
    /* nome_arqPb = ler_str("nome do arquivo de saida Pb");*/
    nome_arqPb = clui_info->o3_file;
  /* Extari a componente Pb a partir de R, G e B */
    COMPOE(R,G,B,PbN,d20,d21,d22,R->linha,R->coluna,unsigned char,float);

    /*printf("\nMatriz PbN :");
    mostra_mat(PbN);*/

    ESCALAR_MAT(PbN,PbN,+,127.5,R->linha,R->coluna,0,0,float,float);

    /*printf("\nMatriz PbN :");
    mostra_mat(PbN);*/

    pbn = (float **)PbN->ptr_mat;
    pb = (unsigned char **)Pb->ptr_mat;
    for(i = 0; i < R->linha; i++)
```

```
            for(j = 0; j < R->coluna; j++)
              pb[i][j] = ROUND(pbn[i][j]);
/* Grava a matriz da componente Pb */
    grava_mat(Pb,nome_arqPb);

    planePb = (unsigned char *)kmalloc(Pb->coluna*Pb->linha*sizeof(unsigned char));
    for(i = 0; i < Pb->linha; i++){
            memcpy(&planePb[i*Pb->coluna],Pb->ptr_mat[i],sizeof(unsigned char)*Pb->coluna);
    }
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planePb);

    /*for(i = 0; i < Pb->linha; i++){
            for(j = 0; j < Pb->coluna; j++){
              value = (unsigned char)Pb->ptr_mat[i][j];
              kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

    /* printf("\nMatriz Pb :");
    mostra_mat(Pb);*/

    libera_mat(Pb);
    libera_mat(PbN);

/* -main_library_call_end */

/* -main_after_lib_call */
   libera_mat(R);
   libera_mat(G);
   libera_mat(B);
   /* Step 7: fecha o arquivo out_obj1  */
   if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj1);

   /* Step 8: fecha o arquivo out_obj2 */
   if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj2);

   /* Step 9: fecha o arquivo out_obj3 */
   if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


            kexit(KEXIT_SUCCESS);
}


/*-------------------------------------------------------------
|
| Routine Name: rgbyprpb_usage_additions
|
|     Purpose: Prints usage additions in rgbyprpb_usage routine
|
|      Input: None
|
|     Output: None
|   Written By: ghostwriter -oname rgbyprpb
|       Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/
void rgbyprpb_usage_additions(void)
{
            kfprintf(kstderr, "\tConverte as componentes RGB nas componentes YPrPb\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-------------------------------------------------------------
|
| Routine Name: rgbyprpb_free_args
|
|     Purpose: Frees CLUI struct allocated in rgbyprpb_get_args()
|
|      Input: None
|
|     Output: None
|   Written By: ghostwriter -oname rgbyprpb
|       Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/
/* ARGSUSED */
void
```

```
rgbyprpb_free_args(
   kexit_status status,
   kaddr       client_data)
{

        /* do the wild and free thing */
        if (clui_info != NULL)
        {
                kfree_and_NULL(clui_info->i1_file);
                kfree_and_NULL(clui_info->i2_file);
                kfree_and_NULL(clui_info->i3_file);
                kfree_and_NULL(clui_info->o1_file);
                kfree_and_NULL(clui_info->o2_file);
                kfree_and_NULL(clui_info->o3_file);
                kfree_and_NULL(clui_info);
        }

/* -free_handler_additions */
/* -free_handler_additions_end */
}



1
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for rgbyprpb
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>         Date: Jul 16, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _rgbyprpb_h_
#define _rgbyprpb_h_

    /*---------------------------------------*
     |     #includes
     ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/library/coreslib/src/maniparq.h"
/* -include_includes_end */


    /*---------------------------------------*
     |     #defines
     ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/rgbyprpb/uis/rgbyprpb.pane"
    /*---------------------------------------*
     |     typedefs
     ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente R (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente R FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente R FLAG */

/*
 * Arquivo de entrada da componente G (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente G FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente G FLAG */

/*
 * Arquivo de entrada da componente B (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente B FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente B FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente Y (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente Y FILENAME */
int          o1_flag;   /* Arquivo de saida da componente Y FLAG */

/*
 * Arquivo de saida da componente Pb (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente Pb FILENAME */
int          o2_flag;   /* Arquivo de saida da componente Pb FLAG */

/*
 * Arquivo de saida da componente Pb (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de saida da componente Pb FILENAME */
int          o3_flag;   /* Arquivo de saida da componente Pb FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


      /*---------------------------------------*
      |     global variable declarations
      ---------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


      /*---------------------------------------*
      |     macros
      ---------------------------------------*/

/* -include_macros */
/* -include_macros_end */

      /*---------------------------------------*
      |     routine definitions
      ---------------------------------------*/

void main PROTO((int, char **));
void rgbyprpb_get_args PROTO((kform *));
void rgbyprpb_usage_additions PROTO((void));
void rgbyprpb_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for yuv_rgb
  >>>>
  >>>> Private:
  >>>>     main
  >>>>
  >>>> Static:
  >>>> Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "yuv_rgb.h"

clui_info_struct *clui_info = NULL;

/*-----------------------------------------------------------
|
| Routine Name: main() - Converte as componentes YUV em RGB
|
|     Purpose: main program for yuv_rgb
|
|      Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente Y}
|                       int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente U}
|                       int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente V}
|                       int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|
|                       int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente R}
|                       int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente G}
|                       int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de saida da componente B}
|                       int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|     Output:
|    Returns:
|
|  Written By:
|       Date: Jul 13, 1998
| Modifications:
|
-----------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "yuv_rgb",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*U,*V,*YN,*UN,*VN,*RN,*GN,*BN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b;
   float **rn,**gn,**bn,**un,**vn;
   int i,j,n,valor,linha,coluna;
   float val;
  /* unsigned char value;*/
   unsigned char *planeR=NULL;
   unsigned char *planeG=NULL;
   unsigned char *planeB=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqR,*nome_arqG,*nome_arqB;

/* Valores da matriz de convercao PAL_M -> RGB */

   d00=1.0;   d01=-0.00004223; d02= 1.402;    /* NORMALIZADO */
```

```
  d10=1.0;  d11=-0.39461;  d12=-0.71401;  /* NORMALIZADO */
  d20=1.0;  d21= 2.032;    d22=-0.000592; /* NORMALIZADO */

/* -main_variable_list_end */

        khoros_initialize(argc, argv, "CORES");
        kexit_handler(yuv_rgb_free_args, NULL);

/* -main_get_args_call */
        kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "yuv_rgb",
                yuv_rgb_usage_additions, yuv_rgb_get_args,
                yuv_rgb_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */

        /* Passo 1: cria o primeiro arquivo de saida */
    if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
     {
      kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
      kexit(KEXIT_FAILURE);
     }

        /* Passo 2: cria o segundo arquivo de saida */
    if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
     {
      kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
      kexit(KEXIT_FAILURE);
     }

        /* Passo 3: cria o terceiro arquivo de saida */
    if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
     {
      kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
      kexit(KEXIT_FAILURE);
     }

    linha =  clui_info->int1_int;
    coluna =  clui_info->int2_int;

    /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
    kpds_create_value(out_obj1);
        kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
        kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

    /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
    kpds_create_value(out_obj2);
        kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
        kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

    /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
    kpds_create_value(out_obj3);
        kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
        kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);

/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente Y no formato INT */
  do {
   /* nome_arq = ler_str("nome do arquivo de entrada da componente Y");
    linha = ler_int("numero de linhas",1,512);
    coluna = ler_int("numero de colunas",1,1024);*/
    nome_arq = clui_info->i1_file;
    Y = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!Y);

  printf("\nMatriz Y :");
  mostra_mat(Y);

/*************************************************************************/
/* Abre e ler um arquivo da componente U no formato INT */
  do {
   /* nome_arq = ler_str("nome do arquivo de entrada da componente U");*/
    nome_arq = clui_info->i2_file;
    U = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!U);
/*
  printf("\nMatriz U :");
  mostra_mat(U);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente V no formato INT */
  do {
   /*nome_arq = ler_str("nome do arquivo de entrada da componente V");*/
    nome_arq = clui_info->i3_file;
    V = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!V);
/*
  printf("\nMatriz V :");
  mostra_mat(V);
*/
/*************************************************************************/
/* Converte as componentes lidas de unsigned char para float */
    YN = aloca_mat(Y->linha,Y->coluna,sizeof(float));
    ESCALAR_MAT(Y,YN,*,1.0,linha,coluna,0,0,unsigned char,float);
/*
    printf("\nMatriz YN :");
```

```
    mostra_mat(YN);
*/
    libera_mat(Y);
    UN = aloca_mat(U->linha,U->coluna,sizeof(float));
    ESCALAR_MAT(U,UN,-,127.5,linha,coluna,0,0,unsigned char,float);
    un = (float **)UN->ptr_mat;
    for(i = 0; i < UN->linha; i++) {
            for(j = 0; j < UN->coluna; j++) {
              val = un[i][j];
              if((abs(val)) <= 0.5)  val = 0.0;
              un[i][j] = ROUND(val);
             }
    }
/*
    printf("\nMatriz UN :");
    mostra_mat(UN);
*/
    libera_mat(U);
    VN = aloca_mat(V->linha,V->coluna,sizeof(float));
    ESCALAR_MAT(V,VN,-,127.5,linha,coluna,0,0,unsigned char,float);
    vn = (float **)VN->ptr_mat;
    for(i = 0; i < VN->linha; i++) {
            for(j = 0; j < VN->coluna; j++) {
              val = vn[i][j];
              if((abs(val)) <= 0.5)  val = 0.0;
              vn[i][j] = ROUND(val);
             }
    }
/*
    printf("\nMatriz VN :");
    mostra_mat(VN);
*/
    libera_mat(V);
/*********************************************************************/
/*********************************************************************/
/* Aloca espaco para R, abre arquivo, extrai a componante R e grava
  no formato UNSIGNED CHAR */
    RN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    R = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqR = ler_str("nome do arquivo de saida R");*/
    nome_arqR = clui_info->o1_file;
  /* Extrai a componente R a partir de Y, U e V */
    COMPOE(YN,UN,VN,RN,d00,d01,d02,linha,coluna,float,float);
/*
    printf("\nMatriz RN :");
    mostra_mat(RN);
*/
    rn = (float **)RN->ptr_mat;
    r = (unsigned char **)R->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(rn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              r[i][j] = valor;
             }
    }
/* Grava a matriz da componente R */
    grava_mat(R,nome_arqR);

    planeR = (unsigned char *)kmalloc(R->coluna*R->linha*sizeof(unsigned char));
    for(i = 0; i < R->linha; i++){
             memcpy(&planeR[i*R->coluna],R->ptr_mat[i],sizeof(unsigned char)*R->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeR);

    /*for(i = 0; i < R->linha; i++){
            for(j = 0; j < R->coluna; j++){
              value = (unsigned char)R->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/

    printf("\nMatriz R :");
    mostra_mat(R);

    libera_mat(R);
    libera_mat(RN);
/*********************************************************************/
/* Aloca espaco para G, abre arquivo, extrai a componante G e grava
  no formato UNSIGNED CHAR */
    GN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    G = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
    /*nome_arqG = ler_str("nome do arquivo de saida G");*/
    nome_arqG = clui_info->o2_file;
  /* Extrai a componente G a partir de Y, U e V */
    COMPOE(YN,UN,VN,GN,d10,d11,d12,linha,coluna,float,float);
/*
    printf("\nMatriz GN :");
    mostra_mat(GN);
*/
    gn = (float **)GN->ptr_mat;
    g = (unsigned char **)G->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor =  ROUND(gn[i][j]);    /* (int)(gn[i][j]);*/
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
```

```
                g[i][j] = valor;
              }
        }
/* Grava a matriz da componente G */
    grava_mat(G,nome_arqG);

    planeG = (unsigned char *)kmalloc(G->coluna*G->linha*sizeof(unsigned char));
    for(i = 0; i < G->linha; i++){
            memcpy(&planeG[i*G->coluna],G->ptr_mat[i],sizeof(unsigned char)*G->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeG);

   /* for(i = 0; i < G->linha; i++){
            for(j = 0; j < G->coluna; j++){
              value = (unsigned char)G->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz G :");
    mostra_mat(G);
*/
    libera_mat(G);
    libera_mat(GN);
/*********************************************************************/
/* Aloca espaco para B, abre arquivo, extrai a componante B e grava
  no formato UNSIGNED CHAR */
    BN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    B = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqB = ler_str("nome do arquivo de saida B");*/
    nome_arqB = clui_info->o3_file;
  /* Extrai a componente B a partir de Y, U e V */
    COMPOE(YN,UN,VN,BN,d20,d21,d22,linha,coluna,float,float);
/*
    printf("\nMatriz BN :");
    mostra_mat(BN);
*/
    bn = (float **)BN->ptr_mat;
    b = (unsigned char **)B->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor =  ROUND(bn[i][j]);   /* (int)(bn[i][j]); */
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              b[i][j] = valor;
            }
    }

/* Grava a matriz da componente B */
    grava_mat(B,nome_arqB);

    planeB = (unsigned char *)kmalloc(B->coluna*B->linha*sizeof(unsigned char));
    for(i = 0; i < B->linha; i++){
            memcpy(&planeB[i*B->coluna],B->ptr_mat[i],sizeof(unsigned char)*B->coluna);
    }
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeB);

    /*for(i = 0; i < B->linha; i++){
            for(j = 0; j < B->coluna; j++){
              value = (unsigned char)B->ptr_mat[i][j];
              kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz B :");
    mostra_mat(B);
*/
    libera_mat(B);
    libera_mat(BN);

/* -main_library_call_end */

/* -main_after_lib_call */
   libera_mat(YN);
   libera_mat(UN);
   libera_mat(VN);

   /* Step 7: fecha o arquivo out_obj1  */
   if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj1);

   /* Step 8: fecha o arquivo out_obj2 */
   if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj2);

   /* Step 9: fecha o arquivo out_obj3 */
   if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
```

```
            }
    kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


            kexit(KEXIT_SUCCESS);
}


/*-------------------------------------------------------------
|
| Routine Name: yuv_rgb_usage_additions
|
|     Purpose: Prints usage additions in yuv_rgb_usage routine
|
|       Input: None
|
|      Output: None
|   Written By: ghostwriter -oname yuv_rgb
|        Date: Jul 13, 1998
| Modifications:
|
-------------------------------------------------------------*/
void yuv_rgb_usage_additions(void)
{
            kfprintf(kstderr, "\tConverte as componentes YUV em RGB\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-------------------------------------------------------------
|
| Routine Name: yuv_rgb_free_args
|
|     Purpose: Frees CLUI struct allocated in yuv_rgb_get_args()
|
|       Input: None
|
|      Output: None
|   Written By: ghostwriter -oname yuv_rgb
|        Date: Jul 13, 1998
| Modifications:
|
-------------------------------------------------------------*/
/* ARGSUSED */
void
yuv_rgb_free_args(
    kexit_status status,
    kaddr       client_data)
{

            /* do the wild and free thing */
            if (clui_info != NULL)
            {
                        kfree_and_NULL(clui_info->i1_file);
                        kfree_and_NULL(clui_info->i2_file);
                        kfree_and_NULL(clui_info->i3_file);
                        kfree_and_NULL(clui_info->o1_file);
                        kfree_and_NULL(clui_info->o2_file);
                        kfree_and_NULL(clui_info->o3_file);
                        kfree_and_NULL(clui_info);
            }

/* -free_handler_additions */
/* -free_handler_additions_end */
}



1
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for yuv_rgb
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>        Date: Jul 13, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _yuv_rgb_h_
#define _yuv_rgb_h_

    /*-------------------------------------*
     |     #includes
     -------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "maniparq.h"
/* -include_includes_end */


    /*-------------------------------------*
     |     #defines
     -------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/yuv_rgb/uis/yuv_rgb.pane"
    /*-------------------------------------*
     |     typedefs
     -------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente Y (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente Y FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente Y FLAG */

/*
 * Arquivo de entrada da componente U (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente U FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente U FLAG */

/*
 * Arquivo de entrada da componente V (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente V FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente V FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente R (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente R FILENAME */
int          o1_flag;   /* Arquivo de saida da componente R FLAG */

/*
 * Arquivo de saida da componente G (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente G FILENAME */
int          o2_flag;   /* Arquivo de saida da componente G FLAG */

/*
 * Arquivo de saida da componente B (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de saida da componente B FILENAME */
int          o3_flag;   /* Arquivo de saida da componente B FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


      /*---------------------------------------*
       |     global variable declarations
       ---------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


      /*---------------------------------------*
       |     macros
       ---------------------------------------*/

/* -include_macros */
/* -include_macros_end */

      /*---------------------------------------*
       |     routine definitions
       ---------------------------------------*/

void main PROTO((int, char **));
void yuv_rgb_get_args PROTO((kform *));
void yuv_rgb_usage_additions PROTO((void));
void yuv_rgb_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for yiq_rgb
  >>>>
  >>>>  Private:
  >>>>     main
  >>>>
  >>>>  Static:
  >>>>  Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "yiq_rgb.h"

clui_info_struct *clui_info = NULL;

/*-------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes YIQ nas componentes RGB
|
|     Purpose: main program for yiq_rgb
|
|      Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente Y}
|                       int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente I}
|                       int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente Q}
|                       int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|
|                       int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente R}
|                       int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente G}
|                       int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de saida da componente B}
|                       int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|     Output:
|    Returns:
|
|  Written By:
|       Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "yuv_rgb",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*I,*Q,*YN,*IN,*QN,*RN,*GN,*BN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b;
   float **rn,**gn,**bn,**in,**qn;
   int i,j,n,linha,coluna,valor;
   float val;
  /* unsigned char value;*/
   unsigned char *planeR=NULL;
   unsigned char *planeG=NULL;
   unsigned char *planeB=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqR,*nome_arqG,*nome_arqB;

/* Valores da matriz de conversao NTSC->RGB */

   d00=1.0;  d01= 1.1391;  d02= 0.6484;  /* NORMALIZADO */
```

```
   d10=1.0;  d11=-0.32369; d12=-0.67665; /* NORMALIZADO */
   d20=1.0;  d21=-1.321;   d22= 1.7835;  /* NORMALIZADO */
/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(yiq_rgb_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "yiq_rgb",
                        yiq_rgb_usage_additions, yiq_rgb_get_args,
                        yiq_rgb_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
            kpds_create_value(out_obj3);
                  kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
                  kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);

/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente Y no formato unsigned char */
  do {
    /* nome_arq = ler_str("nome do arquivo de entrada da componente Y");
     linha = ler_int("numero de linhas",1,512);
     coluna = ler_int("numero de colunas",1,1024);*/
     nome_arq = clui_info->i1_file;
     Y = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!Y);
/*
  printf("\nMatriz Y :");
  mostra_mat(Y);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente I no formato unsigned char */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente I");*/
     nome_arq = clui_info->i2_file;
     I = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!I);
/*
  printf("\nMatriz I :");
  mostra_mat(I);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente Q no formato unsigned char */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente Q");*/
     nome_arq = clui_info->i3_file;
     Q = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!Q);
/*
  printf("\nMatriz Q :");
  mostra_mat(Q);
*/
/*************************************************************************/
/* Converte as componentes lidas de unsigned char para float */
     YN = aloca_mat(Y->linha,Y->coluna,sizeof(float));
     ESCALAR_MAT(Y,YN,*,1.0,Y->linha,Y->coluna,0,0,unsigned char,float);
/*
     printf("\nMatriz YN :");
     mostra_mat(YN);
*/
```

```
    libera_mat(Y);
    IN = aloca_mat(I->linha,I->coluna,sizeof(float));
    ESCALAR_MAT(I,IN,-,127.5,I->linha,I->coluna,0,0,unsigned char,float);
/*
    printf("\nMatriz IN :");
    mostra_mat(IN);
*/
    in = (float **)IN->ptr_mat;
    for(i = 0; i < IN->linha; i++) {
            for(j = 0; j < IN->coluna; j++) {
              val = in[i][j];
              if((abs(val)) <= 0.5)  val = 0.0;
              in[i][j] = ROUND(val);
             }
     }
/*
    printf("\nMatriz IN :");
    mostra_mat(IN);
*/
    libera_mat(I);
    QN = aloca_mat(Q->linha,Q->coluna,sizeof(float));
    ESCALAR_MAT(Q,QN,-,127.5,Q->linha,Q->coluna,0,0,unsigned char,float);
/*
    printf("\nMatriz QN :");
    mostra_mat(QN);
*/
    qn = (float **)QN->ptr_mat;
    for(i = 0; i < QN->linha; i++) {
            for(j = 0; j < QN->coluna; j++) {
              val = qn[i][j];
              if((abs(val)) <= 0.5)  val = 0.0;
              qn[i][j] = ROUND(val);
             }
     }
/*
    printf("\nMatriz QN :");
    mostra_mat(QN);
*/
    libera_mat(Q);
/*********************************************************************/
/*********************************************************************/
/* Aloca espaco para R, abre arquivo, extrai a componante R e grava
  no formato UNSIGNED CHAR */
    RN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    R = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqR = ler_str("nome do arquivo de saida R");*/
    nome_arqR = clui_info->o1_file;
  /* Extrai a componente R a partir de Y, I e Q */
    COMPOE(YN,IN,QN,RN,d00,d01,d02,YN->linha,YN->coluna,float,float);
    rn = (float **)RN->ptr_mat;
    r = (unsigned char **)R->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(rn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              r[i][j] = valor;
             }
     }
/* Grava a matriz da componente R */
    grava_mat(R,nome_arqR);

    planeR = (unsigned char *)kmalloc(R->coluna*R->linha*sizeof(unsigned char));
    for(i = 0; i < R->linha; i++){
             memcpy(&planeR[i*R->coluna],R->ptr_mat[i],sizeof(unsigned char)*R->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeR);

    /*for(i = 0; i < R->linha; i++){
            for(j = 0; j < R->coluna; j++){
              value = (unsigned char)R->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz R :");
    mostra_mat(R);
*/
    libera_mat(R);
    libera_mat(RN);
/*********************************************************************/
/* Aloca espaco para G, abre arquivo, extrai a componante G e grava
  no formato UNSIGNED CHAR */
    GN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    G = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
    /*nome_arqG = ler_str("nome do arquivo de saida G");*/
    nome_arqG = clui_info->o2_file;
  /* Extrai a componente G a partir de Y, I e Q */
    COMPOE(YN,IN,QN,GN,d10,d11,d12,YN->linha,YN->coluna,float,float);
    gn = (float **)GN->ptr_mat;
    g = (unsigned char **)G->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(gn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              g[i][j] = valor;
             }
```

```
    }

/* Grava a matriz da componente G */
    grava_mat(G,nome_arqG);

    planeG = (unsigned char *)kmalloc(G->coluna*G->linha*sizeof(unsigned char));
    for(i = 0; i < G->linha; i++){
            memcpy(&planeG[i*G->coluna],G->ptr_mat[i],sizeof(unsigned char)*G->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeG);

    /*for(i = 0; i < G->linha; i++){
            for(j = 0; j < G->coluna; j++){
              value = (unsigned char)G->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz G :");
    mostra_mat(G);
*/
    libera_mat(G);
    libera_mat(GN);
/*********************************************************************/
/* Aloca espaco para B, abre arquivo, extrai a componante B e grava
  no formato UNSIGNED CHAR */
    BN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    B = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqB = ler_str("nome do arquivo de saida B");*/
    nome_arqB = clui_info->o3_file;
  /* Extrai a componente B a partir de Y, I e Q */
    COMPOE(YN,IN,QN,BN,d20,d21,d22,YN->linha,YN->coluna,float,float);
    bn = (float **)BN->ptr_mat;
    b = (unsigned char **)B->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(bn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              b[i][j] = valor;
            }
    }

/* Grava a matriz da componente B */
    grava_mat(B,nome_arqB);

    planeB = (unsigned char *)kmalloc(B->coluna*B->linha*sizeof(unsigned char));
    for(i = 0; i < B->linha; i++){
            memcpy(&planeB[i*B->coluna],B->ptr_mat[i],sizeof(unsigned char)*B->coluna);
    }
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeB);

    /* for(i = 0; i < B->linha; i++){
            for(j = 0; j < B->coluna; j++){
              value = (unsigned char)B->ptr_mat[i][j];
              kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz B :");
    mostra_mat(B);
*/
    libera_mat(B);
    libera_mat(BN);
/* -main_library_call_end */

/* -main_after_lib_call */
   libera_mat(YN);
   libera_mat(IN);
   libera_mat(QN);

   /* Step 7: fecha o arquivo out_obj1  */
   if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj1);

   /* Step 8: fecha o arquivo out_obj2 */
   if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj2);

   /* Step 9: fecha o arquivo out_obj3 */
   if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj3);

/* -main_after_lib_call_end */
```

```
        kexit(KEXIT_SUCCESS);
}


/*-------------------------------------------------------------
|
| Routine Name: yiq_rgb_usage_additions
|
|     Purpose: Prints usage additions in yiq_rgb_usage routine
|
|       Input: None
|
|      Output: None
|   Written By: ghostwriter -oname yiq_rgb
|        Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/
void yiq_rgb_usage_additions(void)
{
        kfprintf(kstderr, "\tConverte as componentes YIQ nas componentes RGB\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-------------------------------------------------------------
|
| Routine Name: yiq_rgb_free_args
|
|     Purpose: Frees CLUI struct allocated in yiq_rgb_get_args()
|
|       Input: None
|
|      Output: None
|   Written By: ghostwriter -oname yiq_rgb
|        Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/
/* ARGSUSED */
void
yiq_rgb_free_args(
   kexit_status status,
   kaddr       client_data)
{

        /* do the wild and free thing */
        if (clui_info != NULL)
                {
                        kfree_and_NULL(clui_info->i1_file);
                        kfree_and_NULL(clui_info->i2_file);
                        kfree_and_NULL(clui_info->i3_file);
                        kfree_and_NULL(clui_info->o1_file);
                        kfree_and_NULL(clui_info->o2_file);
                        kfree_and_NULL(clui_info->o3_file);
                        kfree_and_NULL(clui_info);
                }

/* -free_handler_additions */
/* -free_handler_additions_end */
}


1
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for yiq_rgb
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>         Date: Jul 16, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _yiq_rgb_h_
#define _yiq_rgb_h_

    /*---------------------------------------*
    |     #includes
    ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/library/coreslib/src/maniparq.h"
/* -include_includes_end */


    /*---------------------------------------*
    |     #defines
    ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/yiq_rgb/uis/yiq_rgb.pane"
    /*---------------------------------------*
    |     typedefs
    ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente Y (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente Y FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente Y FLAG */

/*
 * Arquivo de entrada da componente I (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente I FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente I FLAG */

/*
 * Arquivo de entrada da componente Q (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente Q FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente Q FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente R (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente R FILENAME */
int          o1_flag;   /* Arquivo de saida da componente R FLAG */

/*
 * Arquivo de saida da componente G (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente G FILENAME */
int          o2_flag;   /* Arquivo de saida da componente G FLAG */

/*
 * Arquivo de saida da componente B (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de saida da componente B FILENAME */
int          o3_flag;   /* Arquivo de saida da componente B FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


    /*----------------------------------------*
    |     global variable declarations
    ----------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


    /*----------------------------------------*
    |     macros
    ----------------------------------------*/

/* -include_macros */
/* -include_macros_end */

    /*----------------------------------------*
    |     routine definitions
    ----------------------------------------*/

void main PROTO((int, char **));
void yiq_rgb_get_args PROTO((kform *));
void yiq_rgb_usage_additions PROTO((void));
void yiq_rgb_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for ycrcbrgb
  >>>>
  >>>>  Private:
  >>>>     main
  >>>>
  >>>>  Static:
  >>>>  Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "ycrcbrgb.h"

clui_info_struct *clui_info = NULL;

/*-------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes YCrCb nas componentes RGB
|
|     Purpose: main program for ycrcbrgb
|
|      Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente Y}
|                       int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente Cr}
|                       int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente Cb}
|                       int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|                       int clui_info->int2_int; {tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente R}
|                       int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente G}
|                       int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de saida da componente B}
|                       int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|     Output:
|    Returns:
|
|  Written By:
|       Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "yuv_rgb",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*Cr,*Cb,*YN,*CrN,*CbN,*RN,*GN,*BN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b;
   float **rn,**gn,**bn;
   int i,j,n,valor,linha,coluna;
  /* unsigned char value;*/
   unsigned char *planeR=NULL;
   unsigned char *planeG=NULL;
   unsigned char *planeB=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqR,*nome_arqG,*nome_arqB;

/* Valores da matriz de conversao YCrCb->RGB */

   d00=1.1644;  d01= 1.5901;   d02=-0.0028683; /* NORMALIZADO */
   d10=1.1644;  d11=-0.815;    d12=-0.39;     /* NORMALIZADO */
```

```
   d20=1.1644;  d21=-0.007099;  d22= 2.0165;   /* NORMALIZADO */
/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(ycrcbrgb_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "ycrcbrgb",
                        ycrcbrgb_usage_additions, ycrcbrgb_get_args,
                        ycrcbrgb_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj3);
            kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);

/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente Y no formato unsigned char */
  do {
     /*nome_arq = ler_str("nome do arquivo de entrada da componente Y");
     linha = ler_int("numero de linhas",1,512);
     coluna = ler_int("numero de colunas",1,1024);*/
     nome_arq = clui_info->i1_file;
     Y = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!Y);
/*
  printf("\nMatriz Y :");
  mostra_mat(Y);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente Cr no formato unsigned char */
  do {
     /*nome_arq = ler_str("nome do arquivo de entrada da componente Cr");*/
     nome_arq = clui_info->i2_file;
     Cr = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!Cr);
/*
  printf("\nMatriz Cr :");
  mostra_mat(Cr);
*/
/*************************************************************************/
/* Abre e ler um arquivo da componente Cb no formato unsigned char */
  do {
   /*  nome_arq = ler_str("nome do arquivo de entrada da componente Cb");*/
     nome_arq = clui_info->i3_file;
     Cb = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
   } while(!Cb);
/*
  printf("\nMatriz Cb :");
  mostra_mat(Cb);
*/
/*************************************************************************/
/* Converte as componentes lidas de unsigned char para float */
     YN = aloca_mat(Y->linha,Y->coluna,sizeof(float));
     ESCALAR_MAT(Y,YN,-,16.0,Y->linha,Y->coluna,0,0,unsigned char,float);
     libera_mat(Y);
     CrN = aloca_mat(Cr->linha,Cr->coluna,sizeof(float));
     ESCALAR_MAT(Cr,CrN,-,128.0,Cr->linha,Cr->coluna,0,0,unsigned char,float);
     libera_mat(Cr);
     CbN = aloca_mat(Cb->linha,Cb->coluna,sizeof(float));
```

```
    ESCALAR_MAT(Cb,CbN,-,128.0,Cb->linha,Cb->coluna,0,0,unsigned char,float);
    libera_mat(Cb);
/*********************************************************************/
/*********************************************************************/
/* Aloca espaco para R, abre arquivo, extrai a componante R e grava
  no formato UNSIGNED CHAR */
    RN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    R = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
    /*nome_arqR = ler_str("nome do arquivo de saida R");*/
    nome_arqR = clui_info->o1_file;
  /* Extrai a componente R a partir de Y, Cr e Cb */
    COMPOE(YN,CrN,CbN,RN,d00,d01,d02,YN->linha,YN->coluna,float,float);
    rn = (float **)RN->ptr_mat;
    r = (unsigned char **)R->ptr_mat;
    for(i = 0; i < YN->linha; i++)  {
            for(j = 0; j < YN->coluna; j++) {
              valor =  ROUND(rn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              r[i][j] = valor;
             }
    }
/* Grava a matriz da componente R */
    grava_mat(R,nome_arqR);

    planeR = (unsigned char *)kmalloc(R->coluna*R->linha*sizeof(unsigned char));
    for(i = 0; i < R->linha; i++){
            memcpy(&planeR[i*R->coluna],R->ptr_mat[i],sizeof(unsigned char)*R->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeR);

    /*for(i = 0; i < R->linha; i++){
            for(j = 0; j < R->coluna; j++){
              value = (unsigned char)R->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz R :");
    mostra_mat(R);
*/
    libera_mat(R);
    libera_mat(RN);
/*********************************************************************/
/* Aloca espaco para G, abre arquivo, extrai a componante G e grava
  no formato UNSIGNED CHAR */
    GN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    G = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqG = ler_str("nome do arquivo de saida G");*/
    nome_arqG = clui_info->o2_file;
  /* Extrai a componente G a partir de Y, Cr e Cb */
    COMPOE(YN,CrN,CbN,GN,d10,d11,d12,YN->linha,YN->coluna,float,float);
    gn = (float **)GN->ptr_mat;
    g = (unsigned char **)G->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(gn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              g[i][j] = valor;
             }
     }
/* Grava a matriz da componente G */
    grava_mat(G,nome_arqG);

    planeG = (unsigned char *)kmalloc(G->coluna*G->linha*sizeof(unsigned char));
    for(i = 0; i < G->linha; i++){
            memcpy(&planeG[i*G->coluna],G->ptr_mat[i],sizeof(unsigned char)*G->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeG);

    /*for(i = 0; i < G->linha; i++){
            for(j = 0; j < G->coluna; j++){
              value = (unsigned char)G->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz G :");
    mostra_mat(G);
*/
    libera_mat(G);
    libera_mat(GN);
/*********************************************************************/
/* Aloca espaco para B, abre arquivo, extrai a componante B e grava
  no formato UNSIGNED CHAR */
    BN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    B = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
    /*nome_arqB = ler_str("nome do arquivo de saida B");*/
    nome_arqB = clui_info->o3_file;
  /* Extrai a componente B a partir de Y, Cr e Cb */
    COMPOE(YN,CrN,CbN,BN,d20,d21,d22,YN->linha,YN->coluna,float,float);
    bn = (float **)BN->ptr_mat;
    b = (unsigned char **)B->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(bn[i][j]);
              if(valor < 0) valor = 0;
```

```
                if(valor > 255) valor = 255;
                b[i][j] = valor;
              }
      }
/* Grava a matriz da componente B */
      grava_mat(B,nome_arqB);

      planeB = (unsigned char *)kmalloc(B->coluna*B->linha*sizeof(unsigned char));
      for(i = 0; i < B->linha; i++){
              memcpy(&planeB[i*B->coluna],B->ptr_mat[i],sizeof(unsigned char)*B->coluna);
      }
      kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeB);

      /*for(i = 0; i < B->linha; i++){
              for(j = 0; j < B->coluna; j++){
                value = (unsigned char)B->ptr_mat[i][j];
                kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
              }
      }*/
/*
      printf("\nMatriz B :");
      mostra_mat(B);
*/
      libera_mat(B);
      libera_mat(BN);
/* -main_library_call_end */

/* -main_after_lib_call */
    libera_mat(YN);
    libera_mat(CrN);
    libera_mat(CbN);
    /* Step 7: fecha o arquivo out_obj1  */
    if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
                {
                        kerror(lib,rtn,"Unable to set history on the destination object");
                        kexit(KEXIT_FAILURE);
                }
    kpds_close_object(out_obj1);

    /* Step 8: fecha o arquivo out_obj2 */
    if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
                {
                        kerror(lib,rtn,"Unable to set history on the destination object");
                        kexit(KEXIT_FAILURE);
                }
    kpds_close_object(out_obj2);

    /* Step 9: fecha o arquivo out_obj3 */
    if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
                {
                        kerror(lib,rtn,"Unable to set history on the destination object");
                        kexit(KEXIT_FAILURE);
                }
    kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


        kexit(KEXIT_SUCCESS);
}


/*-----------------------------------------------------------
|
| Routine Name: ycrcbrgb_usage_additions
|
|     Purpose: Prints usage additions in ycrcbrgb_usage routine
|
|      Input: None
|
|     Output: None
|   Written By: ghostwriter -oname ycrcbrgb
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------*/
void ycrcbrgb_usage_additions(void)
{
        kfprintf(kstderr, "\tConverte as componentes YCrCb nas componentes RGB\n");

/* -usage_additions */
/* -usage_additions_end */

}
/*-----------------------------------------------------------
|
| Routine Name: ycrcbrgb_free_args
|
|     Purpose: Frees CLUI struct allocated in ycrcbrgb_get_args()
|
|      Input: None
|
|     Output: None
|   Written By: ghostwriter -oname ycrcbrgb
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------*/
/* ARGSUSED */
```

```
void
ycrcbrgb_free_args(
   kexit_status status,
   kaddr       client_data)
{

            /* do the wild and free thing */
            if (clui_info != NULL)
            {
                        kfree_and_NULL(clui_info->i1_file);
                        kfree_and_NULL(clui_info->i2_file);
                        kfree_and_NULL(clui_info->i3_file);
                        kfree_and_NULL(clui_info->o1_file);
                        kfree_and_NULL(clui_info->o2_file);
                        kfree_and_NULL(clui_info->o3_file);
                        kfree_and_NULL(clui_info);
            }

/* -free_handler_additions */
/* -free_handler_additions_end */
}



1
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for ycrcbrgb
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>         Date: Jul 16, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _ycrcbrgb_h_
#define _ycrcbrgb_h_

    /*---------------------------------------*
    |     #includes
    ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/library/coreslib/src/maniparq.h"
/* -include_includes_end */


    /*---------------------------------------*
    |     #defines
    ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/ycrcbrgb/uis/ycrcbrgb.pane"
    /*---------------------------------------*
    |     typedefs
    ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente Y (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente Y FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente Y FLAG */

/*
 * Arquivo de entrada da componente Cr (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente Cr FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente Cr FLAG */

/*
 * Arquivo de entrada da componente Cb (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente Cb FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente Cb FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* tamanho da imagem em numero de colunas INT */
int          int2_flag; /* tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente R (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente R FILENAME */
int          o1_flag;   /* Arquivo de saida da componente R FLAG */

/*
 * Arquivo de saida da componente G (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente G FILENAME */
int          o2_flag;   /* Arquivo de saida da componente G FLAG */

/*
 * Arquivo de saida da componente B (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de saida da componente B FILENAME */
int          o3_flag;   /* Arquivo de saida da componente B FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


    /*----------------------------------------*
    |     global variable declarations
    ----------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


    /*----------------------------------------*
    |     macros
    ----------------------------------------*/

/* -include_macros */
/* -include_macros_end */

    /*----------------------------------------*
    |     routine definitions
    ----------------------------------------*/

void main PROTO((int, char **));
void ycrcbrgb_get_args PROTO((kform *));
void ycrcbrgb_usage_additions PROTO((void));
void ycrcbrgb_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

```
/*
 * Khoros: $Id$
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: $Id$";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Main program for yprpbrgb
  >>>>
  >>>>  Private:
  >>>>     main
  >>>>
  >>>>  Static:
  >>>>  Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "yprpbrgb.h"

clui_info_struct *clui_info = NULL;

/*-----------------------------------------------------------------
|
| Routine Name: main() - Converte as componentes YPrPb nas componentes RGB
|
|     Purpose: main program for yprpbrgb
|
|      Input:
|                       char *clui_info->i1_file; {Arquivo de entrada da componente Y}
|                       int   clui_info->i1_flag; {TRUE if -i1 specified}
|
|                       char *clui_info->i2_file; {Arquivo de entrada da componente Pr}
|                       int   clui_info->i2_flag; {TRUE if -i2 specified}
|
|                       char *clui_info->i3_file; {Arquivo de entrada da componente Pb}
|                       int   clui_info->i3_flag; {TRUE if -i3 specified}
|
|                       int clui_info->int1_int; {Tamanho da imagem em numero de linhas}
|                       int clui_info->int1_flag; {TRUE if -int1 specified}
|                       int clui_info->int2_int; {Tamanho da imagem em numero de colunas}
|                       int clui_info->int2_flag; {TRUE if -int2 specified}
|
|                       char *clui_info->o1_file; {Arquivo de saida da componente R}
|                       int   clui_info->o1_flag; {TRUE if -o1 specified}
|
|                       char *clui_info->o2_file; {Arquivo de saida da componente G}
|                       int   clui_info->o2_flag; {TRUE if -o2 specified}
|
|                       char *clui_info->o3_file; {Arquivo de Saida da componente B}
|                       int   clui_info->o3_flag; {TRUE if -o3 specified}
|
|     Output:
|    Returns:
|
|  Written By:
|       Date: Jul 16, 1998
| Modifications:
|
-----------------------------------------------------------------*/

void main(
  int  argc,
  char **argv)
{

            kform  *pane;       /* form tree representing *.pane file */
/* -main_variable_list */
   char *lib = "yuv_rgb",*rtn = "main";
   kobject out_obj1 = NULL;
   kobject out_obj2 = NULL;
   kobject out_obj3 = NULL;
   MATRIX *R,*G,*B,*Y,*Pr,*Pb,*YN,*PrN,*PbN,*RN,*GN,*BN;
   float d00,d01,d02,d10,d11,d12,d20,d21,d22;
   unsigned char **r,**g,**b;
   float **rn,**gn,**bn,**prn,**pbn;
   int i,j,n,linha,coluna,valor;
   float val;
   /*unsigned char value;*/
   unsigned char *planeR=NULL;
   unsigned char *planeG=NULL;
   unsigned char *planeB=NULL;
   char tipo;
   char *nome_arq;
   char *nome_arqR,*nome_arqG,*nome_arqB;

/* Valores da matriz de conversao HDTV YPrPb->RGB */

   d00=1.0;  d01= 1.576;  d02= 0.000;  /* NORMALIZADO */
```

```
   d10=1.0;  d11=-0.4766;  d12=-0.2266;  /* NORMALIZADO */
   d20=1.0;  d21=-0.000;   d22= 1.826;   /* NORMALIZADO */
/* -main_variable_list_end */

            khoros_initialize(argc, argv, "CORES");
            kexit_handler(yprpbrgb_free_args, NULL);

/* -main_get_args_call */
            kclui_initialize(PANEPATH, KGEN_KROUTINE, "CORES", "yprpbrgb",
                        yprpbrgb_usage_additions, yprpbrgb_get_args,
                        yprpbrgb_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
            /* Passo 1: cria o primeiro arquivo de saida */
      if ((out_obj1 = kpds_open_output_object(clui_info->o1_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o1_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 2: cria o segundo arquivo de saida */
      if ((out_obj2 = kpds_open_output_object(clui_info->o2_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o2_file);
         kexit(KEXIT_FAILURE);
        }

            /* Passo 3: cria o terceiro arquivo de saida */
      if ((out_obj3 = kpds_open_output_object(clui_info->o3_file)) == KOBJECT_INVALID)
        {
         kerror(lib,rtn,"Cannot open output object %s.\n",clui_info->o3_file);
         kexit(KEXIT_FAILURE);
        }

      linha =  clui_info->int1_int;
      coluna =  clui_info->int2_int;

      /* Passo 4: cria o primeiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj1,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 5: cria o segundo arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj2);
            kpds_set_attribute(out_obj2,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj2,KPDS_VALUE_DATA_TYPE,KUBYTE);

      /* Passo 6: cria o terceiro arquivo de saida com os seguintes parametros */
      kpds_create_value(out_obj3);
            kpds_set_attribute(out_obj3,KPDS_VALUE_SIZE,coluna,linha,1,1,1);
            kpds_set_attribute(out_obj3,KPDS_VALUE_DATA_TYPE,KUBYTE);

/* -main_before_lib_call_end */

/* -main_library_call */
/* Abre e ler um arquivo da componente Y no formato unsigned char */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente Y");
    linha = ler_int("numero de linhas",1,1250);
    coluna = ler_int("numero de colunas",1,2200);*/
    nome_arq = clui_info->i1_file;
    Y = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!Y);
/*
  printf("\nMatriz Y :");
  mostra_mat(Y);
*/
/*********************************************************************/
/* Abre e ler um arquivo da componente Pr no formato unsigned char */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente Pr");*/
    nome_arq = clui_info->i2_file;
    Pr = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!Pr);
/*
  printf("\nMatriz Pr :");
  mostra_mat(Pr);
*/
/*********************************************************************/
/* Abre e ler um arquivo da componente Pb no formato unsigned char */
  do {
    /*nome_arq = ler_str("nome do arquivo de entrada da componente Pb");*/
    nome_arq = clui_info->i3_file;
    Pb = ler_mat(nome_arq,linha,coluna,UNSIGNED_CHAR);
  } while(!Pb);
/*
  printf("\nMatriz Pb :");
  mostra_mat(Pb);
*/
/*********************************************************************/
/* Converte as componentes lidas de unsigned char para float */
    YN = aloca_mat(Y->linha,Y->coluna,sizeof(float));
    ESCALAR_MAT(Y,YN,*,1.0,Y->linha,Y->coluna,0,0,unsigned char,float);
    libera_mat(Y);
    PrN = aloca_mat(Pr->linha,Pr->coluna,sizeof(float));
    ESCALAR_MAT(Pr,PrN,-,127.5,Pr->linha,Pr->coluna,0,0,unsigned char,float);
    libera_mat(Pr);
```

```
    prn = (float **)PrN->ptr_mat;
    for(i = 0; i < PrN->linha; i++) {
            for(j = 0; j < PrN->coluna; j++) {
              val = prn[i][j];
              if((abs(val)) <= 0.5)  val = 0.0;
              prn[i][j] = ROUND(val);
             }
     }
    PbN = aloca_mat(Pb->linha,Pb->coluna,sizeof(float));
    ESCALAR_MAT(Pb,PbN,-,127.5,Pb->linha,Pb->coluna,0,0,unsigned char,float);
    libera_mat(Pb);
    pbn = (float **)PbN->ptr_mat;
    for(i = 0; i < PbN->linha; i++) {
            for(j = 0; j < PbN->coluna; j++) {
              val = pbn[i][j];
              if((abs(val)) <= 0.5)  val = 0.0;
              pbn[i][j] = ROUND(val);
             }
     }
/*************************************************************************/
/*************************************************************************/
/* Aloca espaco para R, abre arquivo, extrai a componante R e grava
  no formato UNSIGNED CHAR */
    RN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    R = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
    /*nome_arqR = ler_str("nome do arquivo de saida R");*/
    nome_arqR = clui_info->o1_file;
  /* Extrai a componente R a partir de Y, Pr e Pb */
    COMPOE(YN,PrN,PbN,RN,d00,d01,d02,YN->linha,YN->coluna,float,float);
    rn = (float **)RN->ptr_mat;
    r = (unsigned char **)R->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(rn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              r[i][j] = valor;
             }
     }
/* Grava a matriz da componente R */
    grava_mat(R,nome_arqR);

    planeR = (unsigned char *)kmalloc(R->coluna*R->linha*sizeof(unsigned char));
    for(i = 0; i < R->linha; i++){
            memcpy(&planeR[i*R->coluna],R->ptr_mat[i],sizeof(unsigned char)*R->coluna);
    }
    kpds_put_data(out_obj1,KPDS_VALUE_PLANE,(kaddr)planeR);

    /*for(i = 0; i < R->linha; i++){
            for(j = 0; j < R->coluna; j++){
              value = (unsigned char)R->ptr_mat[i][j];
              kpds_put_data(out_obj1,KPDS_VALUE_POINT,(kaddr)&value);
             }
    }*/
/*
    printf("\nMatriz R :");
    mostra_mat(R);
*/
    libera_mat(R);
    libera_mat(RN);
/*************************************************************************/
/* Aloca espaco para G, abre arquivo, extrai a componante G e grava
  no formato UNSIGNED CHAR */
    GN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    G = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqG = ler_str("nome do arquivo de saida G");*/
    nome_arqG = clui_info->o2_file;
  /* Extrai a componente G a partir de Y, Pr e Pb */
    COMPOE(YN,PrN,PbN,GN,d10,d11,d12,YN->linha,YN->coluna,float,float);
    gn = (float **)GN->ptr_mat;
    g = (unsigned char **)G->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(gn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              g[i][j] = valor;
             }
     }
/* Grava a matriz da componente G */
    grava_mat(G,nome_arqG);

    planeG = (unsigned char *)kmalloc(G->coluna*G->linha*sizeof(unsigned char));
    for(i = 0; i < G->linha; i++){
            memcpy(&planeG[i*G->coluna],G->ptr_mat[i],sizeof(unsigned char)*G->coluna);
    }
    kpds_put_data(out_obj2,KPDS_VALUE_PLANE,(kaddr)planeG);

   /* for(i = 0; i < G->linha; i++){
            for(j = 0; j < G->coluna; j++){
              value = (unsigned char)G->ptr_mat[i][j];
              kpds_put_data(out_obj2,KPDS_VALUE_POINT,(kaddr)&value);
             }
    }*/
/*
    printf("\nMatriz G :");
    mostra_mat(G);
*/
```

```
    libera_mat(G);
    libera_mat(GN);
/*********************************************************************/
/* Aloca espaco para B, abre arquivo, extrai a componante B e grava
  no formato UNSIGNED CHAR */
    BN = aloca_mat(YN->linha,YN->coluna,sizeof(float));
    B = aloca_mat(YN->linha,YN->coluna,sizeof(unsigned char));
   /* nome_arqB = ler_str("nome do arquivo de saida B");*/
    nome_arqB = clui_info->o3_file;
  /* Extrai a componente B a partir de Y, Pr e Pb */
    COMPOE(YN,PrN,PbN,BN,d20,d21,d22,YN->linha,YN->coluna,float,float);
    bn = (float **)BN->ptr_mat;
    b = (unsigned char **)B->ptr_mat;
    for(i = 0; i < YN->linha; i++) {
            for(j = 0; j < YN->coluna; j++) {
              valor = ROUND(bn[i][j]);
              if(valor < 0) valor = 0;
              if(valor > 255) valor = 255;
              b[i][j] = valor;
             }
     }
/* Grava a matriz da componente B */
    grava_mat(B,nome_arqB);

    planeB = (unsigned char *)kmalloc(B->coluna*B->linha*sizeof(unsigned char));
    for(i = 0; i < B->linha; i++){
            memcpy(&planeB[i*B->coluna],B->ptr_mat[i],sizeof(unsigned char)*B->coluna);
    }
    kpds_put_data(out_obj3,KPDS_VALUE_PLANE,(kaddr)planeB);

    /*for(i = 0; i < B->linha; i++){
            for(j = 0; j < B->coluna; j++){
              value = (unsigned char)B->ptr_mat[i][j];
              kpds_put_data(out_obj3,KPDS_VALUE_POINT,(kaddr)&value);
            }
    }*/
/*
    printf("\nMatriz B :");
    mostra_mat(B);
*/
    libera_mat(B);
    libera_mat(BN);
/* -main_library_call_end */

/* -main_after_lib_call */
   libera_mat(YN);
   libera_mat(PrN);
   libera_mat(PbN);
   /* Step 7: fecha o arquivo out_obj1  */
   if (!kpds_set_attribute(out_obj1, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj1);

   /* Step 8: fecha o arquivo out_obj2 */
   if (!kpds_set_attribute(out_obj2, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj2);

   /* Step 9: fecha o arquivo out_obj3 */
   if (!kpds_set_attribute(out_obj3, KPDS_HISTORY, kpds_history_string()))
            {
                    kerror(lib,rtn,"Unable to set history on the destination object");
                    kexit(KEXIT_FAILURE);
            }
   kpds_close_object(out_obj3);
/* -main_after_lib_call_end */


            kexit(KEXIT_SUCCESS);
}


/*-------------------------------------------------------------
|
| Routine Name: yprpbrgb_usage_additions
|
|    Purpose: Prints usage additions in yprpbrgb_usage routine
|
|      Input: None
|
|    Output: None
|  Written By: ghostwriter -oname yprpbrgb
|       Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/
void yprpbrgb_usage_additions(void)
{
            kfprintf(kstderr, "\tConverte as componentes YPrPb nas componentes RGB\n");

/* -usage_additions */
/* -usage_additions_end */
```

```
}
/*-------------------------------------------------------------
|
| Routine Name: yprpbrgb_free_args
|
|      Purpose: Frees CLUI struct allocated in yprpbrgb_get_args()
|
|        Input: None
|
|       Output: None
|    Written By: ghostwriter -oname yprpbrgb
|         Date: Jul 16, 1998
| Modifications:
|
-------------------------------------------------------------*/
/* ARGSUSED */
void
yprpbrgb_free_args(
   kexit_status status,
   kaddr       client_data)
{

            /* do the wild and free thing */
            if (clui_info != NULL)
            {
                        kfree_and_NULL(clui_info->i1_file);
                        kfree_and_NULL(clui_info->i2_file);
                        kfree_and_NULL(clui_info->i3_file);
                        kfree_and_NULL(clui_info->o1_file);
                        kfree_and_NULL(clui_info->o2_file);
                        kfree_and_NULL(clui_info->o3_file);
                        kfree_and_NULL(clui_info);
            }

/* -free_handler_additions */
/* -free_handler_additions_end */
}
```

```
/*
 * Khoros: $Id$
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose: Include file for yprpbrgb
  >>>>
  >>>>    Written By:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>         Date: Jul 16, 1998
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */


#ifndef _yprpbrgb_h_
#define _yprpbrgb_h_

      /*---------------------------------------*
      |     #includes
      ---------------------------------------*/

#include <cores.h>

/* -include_includes */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/library/coreslib/src/maniparq.h"
/* -include_includes_end */


      /*---------------------------------------*
      |     #defines
      ---------------------------------------*/

/* -include_defines */
/* -include_defines_end */

#define PANEPATH "$CORES/objects/kroutine/yprpbrgb/uis/yprpbrgb.pane"
      /*---------------------------------------*
      |     typedefs
      ---------------------------------------*/

typedef struct _clui_info_struct {

/*
 * Arquivo de entrada da componente Y (required infile)
 */
char        *i1_file;   /* Arquivo de entrada da componente Y FILENAME */
int          i1_flag;   /* Arquivo de entrada da componente Y FLAG */

/*
 * Arquivo de entrada da componente Pr (required infile)
 */
char        *i2_file;   /* Arquivo de entrada da componente Pr FILENAME */
int          i2_flag;   /* Arquivo de entrada da componente Pr FLAG */

/*
 * Arquivo de entrada da componente Pb (required infile)
 */
char        *i3_file;   /* Arquivo de entrada da componente Pb FILENAME */
int          i3_flag;   /* Arquivo de entrada da componente Pb FLAG */

/*
 * Tamanho da imagem em numero de linhas (required integer)
 */
int          int1_int;  /* Tamanho da imagem em numero de linhas INT */
int          int1_flag; /* Tamanho da imagem em numero de linhas FLAG */

/*
 * Tamanho da imagem em numero de colunas (required integer)
 */
int          int2_int;  /* Tamanho da imagem em numero de colunas INT */
int          int2_flag; /* Tamanho da imagem em numero de colunas FLAG */

/*
 * Arquivo de saida da componente R (required outfile)
 */
char        *o1_file;   /* Arquivo de saida da componente R FILENAME */
int          o1_flag;   /* Arquivo de saida da componente R FLAG */

/*
 * Arquivo de saida da componente G (required outfile)
 */
char        *o2_file;   /* Arquivo de saida da componente G FILENAME */
int          o2_flag;   /* Arquivo de saida da componente G FLAG */

/*
 * Arquivo de Saida da componente B (required outfile)
```

```
 */
char        *o3_file;   /* Arquivo de Saida da componente B FILENAME */
int          o3_flag;   /* Arquivo de Saida da componente B FLAG */

} clui_info_struct;

/* -include_typedefs */
/* -include_typedefs_end */


      /*-------------------------------------*
      |     global variable declarations
      -------------------------------------*/

extern clui_info_struct *clui_info;
/* -include_variables */
/* -include_variables_end */


      /*-------------------------------------*
      |     macros
      -------------------------------------*/

/* -include_macros */
/* -include_macros_end */

      /*-------------------------------------*
      |     routine definitions
      -------------------------------------*/

void main PROTO((int, char **));
void yprpbrgb_get_args PROTO((kform *));
void yprpbrgb_usage_additions PROTO((void));
void yprpbrgb_free_args PROTO((kexit_status, kaddr));

/* -include_routines */
/* -include_routines_end */

#endif
```

Programas auxiliares

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"


/*************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                        This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*      Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*   Written By:
*         Date: Nov 15, 1994
*     Verified:
*  Side Effects:
*  Modifications:
*************************************************************/

/*-------------------------------------------------------
|
| Routine Name: foo - short description of foo()
|
|     Purpose: This should be a complete description that anyone
|              could understand;  it should have acceptable grammar
|              and correct spelling.
|
|                        This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|     Returns: TRUE (1) on success, FALSE (0) otherwise
|
|   Written By:
|        Date:
| Modifications:
|
-------------------------------------------------------*/
/*********************************************************************
ABRE_ARQ.C  Abre um arquivo para leitura ou escrita
 *********************************************************************
abre_ler       abre um arquivo para leitura
abre_escreve   abre um arquivo para escrita
 *********************************************************************
 *********************************************************************/


/*********************************************************************
 *********************************************************************
abre_ler   abre um arquivo no modo binariopara leitura e retorna um
               ponteiro para a estrutura alocada pela funcao. Alocacao de
               erro ou tipo inproprio causa exit(1). Um arquivo ruim retorna
```

```
                um ponteiro NULL.

ARQUIVO *abre_ler(char *nome_arq)
*********************************************************************/
ARQUIVO *abre_ler(char *nome_arq)  /* RX ptr p/str do nome do arquivo */
   {     /* retorna um ponteiro para a estrutura do arquivo */
    ARQUIVO *arquivo;   /* informacao de retorno do arquivo */

/* Aloca uma estrutura de dados para o qrquivo */
    arquivo = (ARQUIVO *)malloc(sizeof(ARQUIVO));
    if(!arquivo) {
     printf("\nErro de abertura na alocacao de estrutuda do \
                          arquivo %s \n",nome_arq);
     exit(1);
    }

/* Abre um arquivo para leitura no modo binario se erro, retorna NULL */
    arquivo->ptr_arq = fopen(nome_arq,"r+b");
    if(!arquivo->ptr_arq) {
            printf("\nErro de abertura de %s em abre_ler() \n",nome_arq);
            return(NULL);
    }

/* Aloca e copia a string do nome do arquivo para a estrutura */
    arquivo->nome = (char *)malloc(strlen(nome_arq) + 1);
    if(!arquivo->nome) {
            printf("\nErro na alocacao do nome do arquivo em abre_ler() \n");
            exit(1);
     }
    strcpy(arquivo->nome,nome_arq);

/* Retorna o ponteiro para a estrutura ARQUIVO */
    return(arquivo);
  }
/*********************************************************************
 *********************************************************************
abre_escreve   abre um arquivo no modo binario para escrita e retorna
               um ponteiro para a estrutura alocada pela funcao. Alocacao
               de erro ou tipo inproprio causa exit(1). Um arquivo ruim
               retorna um ponteiro NULL.

ARQUIVO *abre_escreve(char *nome_arq)
*********************************************************************/
ARQUIVO *abre_escreve(char *nome_arq)
  {  /* RX ptr p/str do arq. tipo, n.reg e comp. TX ptr p/struc do arq */
    ARQUIVO *arquivo;    /* informacao de retorno do arquivo */

/* Aloca uma estrutura de dados para o arquivo */
    arquivo = (ARQUIVO *)malloc(sizeof(ARQUIVO));
    if(!arquivo) {
     printf("\nErro de alocacao na abertura da estrura do \
                          arquivo %s \n",nome_arq);
     exit(1);
     }

/* Abre um arquivo para escrita no modo binario se erro, retorna NULL */
    arquivo->ptr_arq = fopen(nome_arq,"wb");
    if(!arquivo->ptr_arq) {
            printf("\nErro de abertura de %s em abre_escreve() \n",nome_arq);
            return(NULL);
     }

/* Aloca e copia a string do nome do arquivo para a estrutura */
    arquivo->nome = (char *)malloc(strlen(nome_arq) + 1);
    if(!arquivo->nome) {
             printf("\nErro na alocacao do nome do arquivo em abre_ler() \n");
             exit(1);
     }
    strcpy(arquivo->nome,nome_arq);

/* Retorna o ponteiro para a estrutura ARQUIVO */
    return(arquivo);
  }
/*********************************************************************/
```

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"

/*****************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*     Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*          Date: Nov 15, 1994
*      Verified:
*  Side Effects:
*  Modifications:
*****************************************************************/

/*-----------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|     Returns: TRUE (1) on success, FALSE (0) otherwise
|
|    Written By:
|          Date:
|  Modifications:
|
-----------------------------------------------------------*/
/*********************************************************************
R_W_REC.C   ler ou escreve um registro em um arquivo aberto no modo
            binario com estrutura ARQUIVO
 *********************************************************************
ler_registro    ler um geristro de um arquivo aberto no modo binario
grava_registro  grava um geristro em um arquivo aberto no modo binario
conv_reg_float  ler um registro char, int dou double e converte para
                float.
 *********************************************************************/

/*********************************************************************
ler_registro            ler um geristro de um arquivo com estrutura ARQUIVO.
                        Provoca exit(1) se ocorrer um erro de leitura ou se a
                        estrutura for invalida. Nao retorna nenhum valor.

void ler_registro(char *ptr_buf, ARQUIVO *arquivo,int coluna,int tamanho)
*********************************************************************/
```

```
void ler_registro(char *ptr_buf, ARQUIVO *arquivo,int coluna,int tamanho)
  {             /* tamanho = numero de bytes */
  int estado;     /* coluna = quantidade de bytes a ser lido */
                          /* ptr_buf = local onde vai ser armazenado */
  if(!arquivo) {
    printf("\nErro na estrutura passada para ler_rec() \n");
    exit(1);
   }
  estado = fread(ptr_buf,tamanho,coluna,arquivo->ptr_arq);
  if(estado != coluna) {
   printf("\nErro na leitura de registro em %s, ",arquivo->nome);
   printf("O tipo gravado eh diferente do tipo pretendido \n");
   exit(1);
   }
 }
/*************************************************************************
 *************************************************************************
grava_registro  grava um geristro de um arquivo com estrutura ARQUIVO.
                        Provoca exit(1) se ocorrer um erro de leitura ou se a
                        estrutura for invalida. Nao retorna nenhum valor.

void grava_registro(char *ptr_reg, ARQUIVO *arquivo,int coluna,int tamanho)
*************************************************************************/
void grava_registro(char *ptr_reg, ARQUIVO *arquivo,int coluna,int tamanho)
  {             /* tamanho = numero de bytes */
  int estado;     /* coluna = quantidade de bytes a ser lido */
                          /* ptr_reg = local de onde vai ser armazenado */
  if(!arquivo) {
    printf("\nErro na estrutura passada para grava_registro() \n");
    exit(1);
   }
  estado = fwrite(ptr_reg,tamanho,coluna,arquivo->ptr_arq);
  if(estado != coluna) {
   printf("\nErro na gravacao de registro em %s \n",arquivo->nome);
   exit(1);
   }
 }
/*************************************************************************
 *************************************************************************
conv_reg_float()  ler um registro do arquivo entrado pelo usuario e
                        converte os dedos lidos para um arranjo float. Retorna
                        um ponteiro para o inicio do arranjo float alocado,
                        representando o registro lido do arquivo.

float *conv_reg_float(ARQUIVO *arquivo,int coluna,unsigned char tipo)
*************************************************************************/
float *conv_reg_float(ARQUIVO *arquivo,int coluna,unsigned char tipo)
  {
  int i, tamanho;  /* coluna = quantidade de bytes a ser lido */
```

```
float *out;     /* tipo = tipo do dado lido CHAR, INT, FLOAT DOUBLE */
float *out_ptr;
static double *buf;
void ler_registro();
long int tamanho_byte;
static long int tamanho_previsto = 0;


/* determina o tamanho de cada elemento do registro (char,int,...) */
switch(tipo) {
                case CHAR:
                case INT:
                   tamanho = sizeof(short int);  /* 2 bytes */
                   break;
                case FLOAT:
                   tamanho = sizeof(float);     /* 4 bytes */
                   break;
                case DOUBLE:
                   tamanho = sizeof(double);    /* 8 bytes */
                   break;
                default:
                  printf("\nTipo nao compativel com o sistema \n");
                  break;
        }

tamanho_byte = (long)coluna*tamanho;  /* tamanho do buffer em bytes */

/* Checa para ver se ja foi alocado buffer de entrada */
if(tamanho_byte != tamanho_previsto) {
        if(tamanho_previsto != 0)
           free(buf);     /* libera o espaco do buffer */
  /* Aloca espaco para o buffer de entrada com precisao double para
        prevenir quanto a perda de alinhamento dos dados */
        buf = (double *)calloc(coluna,tamanho);
        if(!buf) {
           printf("\nErro na alocacao do buffer de entrada \n");
           exit(1);
         }
        tamanho_previsto = tamanho_byte;  /* atualiza a variavel */
 }

/* Aloca o ponteiro de saida somente se for diferrente de FLOAT */
 if(tipo != FLOAT) {
          out = (float *)calloc(coluna,sizeof(float));
          if(!out) {
            printf("\nErro de alocacao em conv_reg_float() \n");
            exit(1);
           }
  }
```

```
  /* Ler um registro de tamanho "coluna" e "tamanho" bytes por elemento
    do arquivo para o buffer */
          ler_registro((char *)buf,arquivo,coluna,tamanho);

   out_ptr = out;   /* aponta para a mesma posicao, out eh fixo,
                                                          out_put eh variavel /*
  /* Realiza a conversao para float */
    switch(tipo) {
                        case CHAR: {
                          char *char_ptr;
                          char_ptr = (char *)buf;  /* aponta para o buffer */
                          for(i = 0; i < coluna; i++)
                                  *out_ptr++ = (float)(*char_ptr++); /* converte */
                         }
                         break;
                        case INT: {
                          int *int_ptr;
                          int_ptr = (int *)buf;
                          for(i = 0; i < coluna; i++)
                                  *out_ptr++ = (float)(*int_ptr++);
                         }
                         break;
                        case FLOAT:
                          out = (float *)buf;
                          tamanho_previsto = 0;
                          break;
                        case DOUBLE: {
                          double *dbl_ptr;
                          dbl_ptr = (double *)buf;
                          for(i = 0; i < coluna; i++)
                             *out_ptr++ = (float)(*dbl_ptr++);
                         }
                         break;
             }
            return(out);

  }
/*************************************************************************/
```

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
   >>>>
   >>>>     Library Routine for coreslib
   >>>>
   >>>>     Private:
   >>>>     Static:
   >>>>     Public:
   >>>>
   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdio.h>
#include <stdlib.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"

/*****************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*     Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*         Date: Nov 15, 1994
*      Verified:
*  Side Effects:
*  Modifications:
*****************************************************************/

/*-----------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|     Returns: TRUE (1) on success, FALSE (0) otherwise
|
|   Written By:
|        Date:
|  Modifications:
|
-----------------------------------------------------------*/
/*************************************************************************
R_W_MAT.C   ler ou escreve uma matriz em um arquivo aberto no modo
            binario com estrutura ARQUIVO
 ************************************************************************
ler_mat     ler uma matriz de um arquivo aberto no modo binario
grava_mat   grava uma matriz em um arquivo aberto no modo binario
mostra_mat  mostra uma matriz no display
 ************************************************************************/

/*************************************************************************
ler_mat   ler uma matriz de um arquivo com estrutura ARQUIVO. Provoca
            exit(1) se ocorrer um erro de leitura ou se a estrutura for
            invalida. Retorna um ponteiro para estrutura MATRIX.

MATRIX *ler_mat(char *nome_arq,int linha,int coluna,unsigned char tipo)
*************************************************************************/
MATRIX *ler_mat(char *nome_arq,int linha,int coluna,unsigned char tipo)
  {
```

```
MATRIX *aloca_mat();  /* alocacao de memoria para matrix */
MATRIX *AAA;                    /* matrix de retorno  */
ARQUIVO *arquivo;               /* estrutura para o arquivo de entrada */
int i,tamanho_elem;
int tamanho;
double *buf;            /* buf para entrada de dados */

arquivo = abre_ler(nome_arq); /*recebe *nome e FILE *ptr_art */
if(!arquivo) {
        return(NULL);  /* caso de arquivo ruim */
  }

/* determina o tamanho da matriz (int, float ou double) */
switch(tipo) {
          case UNSIGNED_CHAR:
                  tamanho = sizeof(unsigned char);
                  tamanho_elem = sizeof(unsigned char);   /* 2 bytes */
                  break;
          case CHAR:
                  tamanho = sizeof(char);
                  tamanho_elem = sizeof(char);   /* 2 bytes */
                  break;
          case INT:
                  tamanho_elem = sizeof(short int);   /* 2 bytes */
                  tamanho = tamanho_elem;
                  break;
          case UNSIGNED_INT:
                  tamanho_elem = sizeof(float);   /* 2 bytes */
                  tamanho = sizeof(unsigned int);
                  break;
          case LONG_INT:
                  tamanho_elem = sizeof(float);   /* 2 bytes */
                  tamanho = sizeof(long int);
                  break;
          case UNSIGNED_LONG:
                  tamanho_elem = sizeof(float);   /* 2 bytes */
                  tamanho = sizeof(unsigned long);
                  break;
          case FLOAT:
                  tamanho = sizeof(float);
                  tamanho_elem = sizeof(float);    /* 4 bytes */
                  break;
          case DOUBLE:
                  tamanho = sizeof(double);
                  tamanho_elem = sizeof(double);   /* 8 bytes */
                  break;
          default:
                  printf("\nTipo nao compativel com o sistema \n");
                  break;
          }


/* Aloca espaco para a matriz "A" retorna com os dados da estrutura
        MATRIX preenchidos (tamanho em bytes,linha,coluna) */
  AAA = aloca_mat(linha,coluna,tamanho_elem);

/* Aloca espaco para buffer de entrada tipo double para o caso
  de perda de alinhamento */
  buf = (double *)calloc(coluna,tamanho);
  if(!buf) {
          printf("\nErro na alocacao de buffer em ler_mat() \n");
          exit(1);
          }

/* Ler cada linha do arquivo e translada para a matriz "A" */
  for(i = 0; i < linha; i++) {
           ler_registro((char *)buf,arquivo,coluna,tamanho);

           switch(tipo) {
             case CHAR:
                     ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,char,int)
                     break;
             case UNSIGNED_CHAR:
           ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,unsigned char,unsigned char)
                     break;
             case INT:
                     ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,int,int)
                     break;
             case UNSIGNED_INT:
                     ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,int,float)
                     break;
             case LONG_INT:
                     ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,long int,float)
                     break;
             case UNSIGNED_LONG:
                    ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,unsigned long,float)
                    break;
             case FLOAT:
                     ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,float,float)
                     break;
             case DOUBLE:
                     ESCALA_VEC(buf,AAA->ptr_mat[i],1,coluna,double,double)
                     break;
             }
         }
/* Libera a estrutura do arquivo e fecha o arquivo */
free(arquivo->nome);
fclose(arquivo->ptr_arq);
```

```
    free((char *)arquivo);

    return(AAA);

  }
/*************************************************************************
 *************************************************************************
grava_mat  grava uma matriz do mesmo tipo dado na estrutura MATRIX (int,
           float ou double) para um arquivo com estrutura ARQUIVO. Provoca
           exit(1) se ocorrer um erro de leitura ou se a estrutura for
           invalida. Retorna um ponteiro para a estrutura ARQUIVO.

ARQUIVO *grava_mat(MATRIX *AAA, char *nome_arq)
*************************************************************************/
ARQUIVO *grava_mat(MATRIX *AAA,char *nome_arq)
  {
   ARQUIVO *arquivo;
   int i;

    /* abre arquivo para escrita, retorna um ponteiro para arquivo ou
            nulo se ocorrer erro */
    arquivo = abre_escreve(nome_arq);
    if(!arquivo) {
            return(NULL);
     }

    /* escreve cada linha da matriz */
     for(i = 0; i < AAA->linha; i++)
              grava_registro(AAA->ptr_mat[i],arquivo,AAA->coluna,AAA->tamanho);

    return(arquivo);

  }
/*************************************************************************
 *************************************************************************
 mostra_mat()  mostra uma matriz no display do monitor usando a funcao
                printf() para mostrar os elementos nos formatos %g ou %d.

void mostra_mat(MATRIX *AAA)
 *************************************************************************/
void mostra_mat(MATRIX *AAA)
    {
            int i, j, col;
            unsigned char **a;
           if(AAA->tamanho == sizeof(short) || AAA->tamanho == sizeof(unsigned char))
              col = 8;  /* 10 colunas por linha para int */
            else
              col = 8;   /* 6 colunas por linha para float e double */

    for(i = 0; i < AAA->linha; i++) {
      if (i > 2) continue;
      for(j = 0; j < AAA->coluna; j++) {
              if(j%col == 0) {  /* nova linha a cada col */
               if(j == 0)  /* inicio da linha  */
                printf("\nLinha%3d:\n",i);
               else
                printf("\n");
              }
              switch(AAA->tamanho) {
                 case sizeof(unsigned char):
              /*          a = (unsigned char **)AAA->ptr_mat;
                          printf("AA[%d][%d] = %7u   ",i,j,a[i][j]);*/
              printf("%6u",((unsigned char **)AAA->ptr_mat)[i][j]);
                          break;
                 case sizeof(short int):
                          printf("%6d",((short **)AAA->ptr_mat)[i][j]);
                          break;
                 case sizeof(float):
                          printf("%9.5g",((float **)AAA->ptr_mat)[i][j]);
                          break;
                 case sizeof(double):
                          printf("%9.5lg",((double **)AAA->ptr_mat)[i][j]);
                          break;
              }
      }
    }
  }
/*************************************************************************/
```

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"


/*****************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*      Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*          Date: Nov 15, 1994
*     Verified:
*  Side Effects:
*  Modifications:
*****************************************************************/

/*---------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|      Returns: TRUE (1) on success, FALSE (0) otherwise
|
|    Written By:
|         Date:
|  Modifications:
|
---------------------------------------------------------*/
/*************************************************************************
MODULA.C - ROTINAS DE MODULACAO DAS COMPONENTES RGB PARA OBTENSAO DO
           SINAL COMPOSTO PAL-M OU NTSC.
*************************************************************************/

#define pi 3.141592654
#define fsc 3.57561149
#define Wsc (2.0*pi*fsc)
#define TH (909/(4.0*fsc))

#define fase1 0.0
#define fase2 (3.0*pi/2.0)
#define fase3 (pi/2.0)
#define fase4 pi
#define coef1 (pi/2.0)  /* espacamento angular entre amostras a 4fsc */
```

```
#define alfa (255.0/(360.0+100.0))  /* y = alfa*x + beta */
#define beta (alfa*100.0)

/*********************************************************************
UV_C4FSC() - Funcao de modulacao em quadratura dos sinais componentes
             U e V para obtero sinal de crominancia Ec para posterior
             aquisicao do sinal composto Em. Recebe os sinais U e V
             no formato float e retorna o sinal Ec em float.

MATRIX *UV_C4FSC(MATRIX *U, MATRIX *V)
*********************************************************************/
MATRIX *UV_C4FSC(MATRIX *U, MATRIX *V)
   {
    MATRIX *aloca_mat();
    MATRIX *C;
    float **u,**v,**c;
    float fase,teta,AM;
    int lin_u, col_u;
    int i,j,IQ,JQ,IR,JR;

    if((U->linha != V->linha) || (U->coluna != V->coluna)) {
            printf("\nErro por diferenca de tamanho entre as matrizes ");
            exit(1);
    }
    /* Atribui os valores de linha e coluna da matrix U */
    lin_u = U->linha;
    col_u = U->coluna;

/* Aloca espaco para o sinal composto C, no normato float  */
    C = aloca_mat(U->linha,U->coluna,sizeof(float));
    c = (float **)C->ptr_mat;
    u = (float **)U->ptr_mat;
    v = (float **)V->ptr_mat;
    for(i = 0; i < lin_u; i++) {
            IQ = (int)((i+1)/8.0);   /* contador de blocos de 8 linhas */
            IR = (i+1) - (8*IQ);     /*indicador de linha a cada 8 linhas */
            for(j = 0; j < col_u; j++) {
              if((IR == 1) || (IR == 4))
                { AM = 1.0;   fase = fase1; }
              if((IR == 2) || (IR >= 7))
                { AM = -1.0;  fase = fase2; }
              if((IR == 3) || (IR == 6))
                { AM = -1.0;  fase = fase3; }
              if((IR == 5) || (IR <= 0))
                { AM = 1.0;   fase = fase4; }
              JQ = (int)((float)j / 4.0);  /* F1 = 4.0 */
              JR = j - ((int)4.0 * JQ);
               teta = fase + coef1*(float)JR;
                c[i][j] = u[i][j]*sin(teta) + AM*v[i][j]*cos(teta);
            }
    }
    return(C);
}
/*********************************************************************/
/*********************************************************************
C4FSC_UV()  Funcao para a detecao sincrona do sinal de crominancia para a
            obtencao dos sinais diferenca de cor U e V. Recebe os sinais UL
            e VL da linha de retardo e o sinal Ec para a realizacao da dete-
            cao sincrona. Retorna os sinais diferenca de cor ULL e VLL no
     formato            float.

void C4FSC_UV(MATRIX *C,MATRIX *UL, MATRIX *VL,MATRIX *ULL,MATRIX *VLL)
*********************************************************************/
/*********************************************************************/
void C4FSC_UV(MATRIX *C,float **ull,float **vll)
 {
  float **ecl;
  float fase,teta,AM,U,V;
  int lin_c, col_c;
  int i,j,IQ,JQ,IR,JR;

   /* Atribui os valores de linha e coluna da matrix U */
   lin_c = C->linha;
   col_c = C->coluna;

   /* Posicao inicial dos ponteiros */
   ecl = (float **)C->ptr_mat;
   for(i = 2; i < lin_c; i++) { /* comeca na linha 3 ate a 512 */
            IQ = (int)((i+1)/8.0);   /* contador de blocos de 8 linhas */
            IR = (i+1) - (8*IQ);     /*indicador de linha a cada 8 linhas */
            for(j = 0; j < col_c; j++) {
              if(j == 0) {   /* testa se eh a 1a amostra na linha */
                U = ecl[i][j] - ecl[i-2][j];
                V = ecl[i][j] - ecl[i-2][j+1];
               }
              if((j != 0) && (j < col_c-1)) {
                U = ecl[i][j] - ecl[i-2][j-1];
                V = ecl[i][j] - ecl[i-2][j+1];
               }
              if(j == col_c-1) {   /* testa se eh a ultima amostra na linha */
                U = ecl[i][j] - ecl[i-2][j];
                V = ecl[i][j] - ecl[i-2][j-1];
               }
             /* Realiza a detecao sincrona do sinal  */
              if((IR == 1) || (IR == 4))
                { AM = 1.0;   fase = fase1; }
              if((IR == 2) || (IR >= 7))
                { AM = -1.0;  fase = fase2; }
```

```
            if((IR == 3) || (IR == 6))
              { AM = -1.0;  fase = fase3; }
            if((IR == 5) || (IR <= 0))
              { AM = 1.0;   fase = fase4; }
            JQ = (int)((float)j / 4.0);   /* F1 = 4.0 */
            JR = j - ((int)4.0 * JQ);
            teta = fase + coef1*(float)JR;
            ull[i][j] = U*sin(teta);    /* ull e vll comecam na linha 3 */
            vll[i][j] = AM*V*cos(teta);
          }
    }
/* Mostra na tela e Grava a matriz do sinal composto EM */
}
/*************************************************************************/
/*********************************************************************
UV_Y() - Funcao de modulacao em quadratura dos sinais componentes
         U e V para obtero sinal de crominancia Ec para posterior
         aquisicao do sinal composto Em. Recebe os sinais U e V
         no formato float e retorna o sinal Ec em float.

MATRIX *UV_Y(MATRIX *ELM,MATRIX *U, MATRIX *V)
*********************************************************************/
/*********************************************************************/
MATRIX *UV_Y(MATRIX *ELM,MATRIX *U, MATRIX *V)
   {
    MATRIX *aloca_mat();
    MATRIX *Y;
    float **u,**v,**elm,**y;
    float fase,teta,AM,EC;
    int lin_u, col_u;
    int i,j,IQ,JQ,IR,JR;

    if((U->linha != V->linha) || (U->coluna != V->coluna)) {
          printf("\nErro por diferenca de tamanho entre as matrizes ");
          exit(1);
    }
    /* Atribui os valores de linha e coluna da matrix U */
    lin_u = U->linha;
    col_u = U->coluna;

/* Aloca espaco para o sinal composto C, no normato float  */
    Y = aloca_mat(U->linha,U->coluna,sizeof(float));
    y = (float **)Y->ptr_mat;
    u = (float **)U->ptr_mat;
    v = (float **)V->ptr_mat;
    elm = (float **)ELM->ptr_mat;
    for(i = 2; i < lin_u; i++) {
          IQ = (int)((i+1)/8.0);   /* contador de blocos de 8 linhas */
          IR = (i+1) - (8*IQ);     /*indicador de linha a cada 8 linhas */
          for(j = 0; j < col_u; j++) {
            if((IR == 1) || (IR == 4))
              { AM = 1.0;   fase = fase1; }
            if((IR == 2) || (IR >= 7))
              { AM = -1.0;  fase = fase2; }
            if((IR == 3) || (IR == 6))
              { AM = -1.0;  fase = fase3; }
            if((IR == 5) || (IR <= 0))
              { AM = 1.0;   fase = fase4; }
            JQ = (int)((float)j / 4.0);   /* F1 = 4.0 */
            JR = j - ((int)4.0 * JQ);
            teta = fase + coef1*(float)JR;
            EC = u[i][j]*sin(teta) + AM*v[i][j]*cos(teta);
            y[i][j] = elm[i][j] - EC;
          }
    }
    return(Y);
}
/*************************************************************************/
/*********************************************************************
NORMALIZA() - Funcao de normalizacao do sinal modulado EM retornando
              o sinal EMN normalizado no formato unsigned char.

MATRIX *NORMALIZA(MATRIX *EM)
*********************************************************************/
/*********************************************************************/
MATRIX *NORMALIZA(MATRIX *EM)
  {
   MATRIX *aloca_mat();
   MATRIX *EMN;
   unsigned char **emn;
   float **em,valor;
   int lin_em, col_em,val;
   int i,j;
    /* Atribui os valores de linha e coluna da matrix U */
    lin_em = EM->linha;
    col_em = EM->coluna;

   EMN = aloca_mat(EM->linha,EM->coluna,sizeof(unsigned char));
   em = (float **)EM->ptr_mat;
   emn = (unsigned char **)EMN->ptr_mat;
   for(i = 0; i < lin_em;i++) {
          for(j = 0; j < col_em; j++) {
            valor = alfa*em[i][j] + beta;
            val = ROUND(valor);
            if(val > 255) val = 255;
            if(val < 0) val = 0;
            emn[i][j] = val;
          }
```

```
            }
  return(EMN);
  }
/*********************************************************************/
/*********************************************************************
DESNORMALIZA() - Funcao de normalizacao do sinal modulado EM retornando
                 o sinal EMN normalizado no formato unsigned char.

MATRIX *DESNORMALIZA(MATRIX *EMM)
*********************************************************************/
/*********************************************************************/
MATRIX *DESNORMALIZA(MATRIX *EMM)
  {
   MATRIX *aloca_mat();
   MATRIX *EMD;
   unsigned char **emm;
   float **emd,valor;
   int lin_em, col_em;
   int i,j;
    /* Atribui os valores de linha e coluna da matrix U */
    lin_em = EMM->linha;
    col_em = EMM->coluna;

   EMD = aloca_mat(EMM->linha,EMM->coluna,sizeof(float));
   emd = (float **)EMD->ptr_mat;
   emm = (unsigned char **)EMM->ptr_mat;
   for(i = 0; i < lin_em;i++) {
              for(j = 0; j < col_em; j++) {
                valor = ((float)emm[i][j] - beta)/alfa;
                emd[i][j] = valor;
               }
             }
  return(EMD);
  }
/*********************************************************************/
```

I

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"


/*****************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*      Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*          Date: Nov 15, 1994
*       Verified:
*  Side Effects:
*  Modifications:
*****************************************************************/

/*---------------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|      Returns: TRUE (1) on success, FALSE (0) otherwise
|
|    Written By:
|          Date:
|  Modifications:
|
---------------------------------------------------------------*/
/*************************************************************************
LER.C  ler string, inteiro e float entrados pelo usuario a partir do
       prompt.
 *************************************************************************
ler_str    ler string entrada pelo usuario a partir do prompt
ler_int    ler inteiro entrado pelo usuario dentro de um range
ler_float  ler float idem

 *************************************************************************
 *************************************************************************/

/*************************************************************************
ler_str  ler uma string entrada pelo usuario a partir do prompt e retorna
```

```
            um ponteiro para a string passada pelo chamador. Indica erro se
            o espaco da string nao for alocado. Limitado em 80 caracter

char *ler_str(char *prompt_str)
*************************************************************************/
char *ler_str(char *prompt_str)  /* recebe uma mensagem texto */
  {
   char *texto;        /* ponteiro da string resultante */

   texto = (char *)malloc(200);
   if(!texto) {
            printf("\nErro de alocacao de string em ler_str() \n");
            exit(1);
   }
   printf("\n\nEntre %s : ",prompt_str);
   gets(texto);

   return(texto);   /* retorna o texto entrado pelo usuario */
  }
/*************************************************************************
 *************************************************************************
ler_int   ler um texto entrado pelo usuario a partir do prompt e retorna
            um inteiro, dentro de um range de valores (inferior e superior)
            passado pelo chamador. Indica erro se o valor passado estiver
            fora do range especificado.

int ler_int(char *prompt_str, int limite_inf, int limite_sup)
*************************************************************************/
int ler_int(char *prompt_str, int limite_inf, int limite_sup)
 {          /* recebe um texto e dois inteiros e retorna um inteiro */
  char *ler_str();       /* rotina de leitura de string */
  int valor, sinal_erro;  /* sinal se ocorrer erro */
  char *cp, *endcp;      /* ponteiro para caracter usado em strtol()  */
  char *str_temp;        /* string temporaria para sprintf()  */

/* testa se ocorre erro nos limites, limite_inf <= limite_sup */
  if(limite_inf > limite_sup) {
   printf("\nErro de limites de entrada, limite_inf > limite_sup \n");
   exit(1);
  }

/* aloca memoria e define uma string temporaria em sprintf() */
  str_temp = (char *)malloc(strlen(prompt_str) + 200);  /* texto + range */
  if(!str_temp) {
   printf("\nErro de alocacao de string em ler_int() \n");
   exit(1);
  }
  sprintf(str_temp, "%s [%d...%d]",prompt_str,limite_inf,limite_sup);

/* ler uma string "valor" e assegura que ela esta dentro do range limite */
  do {
     cp = ler_str(str_temp);  /* manda a nova string para ler_str() */
     valor = (int)strtol(cp,&endcp,10); /* converte char para int longo
                                                    na base 10 apontado por cp e endcp */
     sinal_erro = (cp == endcp) || (*endcp != '\0'); /* ocorre erro se */
     free(cp);      /* libera o ponteiro cp */
    } while(valor < limite_inf || valor > limite_sup || sinal_erro);

/* libera o ponteiro str_temp e retorna um inteiro "valor" */
  free(str_temp);
  return(valor);
 }
/*************************************************************************
 *************************************************************************
ler_float  ler um texto entrado pelo usuario a partir do prompt e retorna
            um double, dentro de um range de valores (inferior e superior)
            passado pelo chamador. Indica erro se o valor passado estiver
            fora do range especificado.

double ler_float(char *prompt_str, int limite_inf, int limite_sup)
*************************************************************************/
double ler_float(char *prompt_str, double limite_inf, double limite_sup)
 {      /* recebe um ponteiro texto e dois double e retorna um double */
  char *ler_str();       /* rotina de leitura de string */
  double valor;          /* valor retornado double */
  int sinal_erro;        /* sinal se ocorrer erro */
  char *cp, *endcp;      /* ponteiro para caracter usado em strtol()  */
  char *str_temp;        /* string temporaria para sprintf()  */

/* testa se ocorre erro nos limites, limite_inf <= limite_sup */
  if(limite_inf > limite_sup) {
   printf("\nErro de limites de entrada, limite_inf > limite_sup \n");
   exit(1);
  }

/* aloca memoria e define uma string temporaria em sprintf() */
  str_temp = (char *)malloc(strlen(prompt_str) + 200);  /* texto + range */
  if(!str_temp) {
   printf("\nErro de alocacao de string em ler_int() \n");
   exit(1);
  }
  sprintf(str_temp, "%s [%1.2g...%1.2g]",prompt_str,limite_inf,limite_sup);

/* ler uma string "valor" e assegura que ela esta dentro do range limite */
  do {
     cp = ler_str(str_temp);  /* manda a nova string para ler_str() */
```

```
    valor = strtod(cp,&endcp); /* converte char para double apontado
                                                    por cp e endcp */
    sinal_erro = (cp == endcp) || (*endcp != '\0'); /* ocorre erro se */
    free(cp);      /* libera o ponteiro cp */
   } while(valor < limite_inf || valor > limite_sup || sinal_erro);

/* libera o ponteiro str_temp e retorna um inteiro "valor" */
  free(str_temp);
  return(valor);
 }
/*************************************************************************/
```

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"


/*****************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*      Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*          Date: Nov 15, 1994
*      Verified:
*  Side Effects:
*  Modifications:
*****************************************************************/

/*---------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|      Returns: TRUE (1) on success, FALSE (0) otherwise
|
|    Written By:
|         Date:
|  Modifications:
|
---------------------------------------------------------*/
/*********************************************************************
ALOC_MAT  aloca e libera espaco de memoria para matriz
 *********************************************************************
aloca_mat    aloca espaco para matriz, retornando um ponteiro para
               a estrutura MATRIX
libera_mat   libera o espaco de memoria e o ponteiro da estrutura
               MATRIX
*********************************************************************
*********************************************************************/

/*********************************************************************
 *********************************************************************
aloca_mat  Aloca espaco para uma matriz de tamanho (linha x coluna)
             e tipo dado pelo tamanho do elemento, podendo ser do tipo
```

```
        int, float ou double (2, 4 ou 8 bytes). Provoca exit(1)
        caso nao possa ser alocado espaco para a mesma.

MATRIX *aloca_mat(unsigned int linha,unsigned int coluna,int tamanho)
*****************************************************************/
MATRIX *aloca_mat(unsigned int linha,unsigned int coluna,int tamanho)
        {
        int i;
        MATRIX *AA;   /* estrutura matriz de retorno */

        /*Aloca espaco para a estrutura da matriz "A" */
        AA = (MATRIX *)calloc(1,sizeof(MATRIX));
        if(!AA) {
          printf("\nErro na alocacao de espaco para a estrutura \n");
          exit(1);
         }

        /*Preenche os dados da estrutura de "AA" com os dados recebidos */
        AA->linha = linha;
        AA->coluna = coluna;
        AA->tamanho = tamanho;

        /*Alocacao de espaco para os dados da matriz, com o tamanho de
          cada elemento podendo ser short, float ou double */
        switch(tamanho) {
                case sizeof(unsigned char): { /* matriz de char ou unsigned */
                  unsigned char **char_mat;
         char_mat = (unsigned char **)calloc(linha,sizeof(unsigned char *));
                          /*Aloca ponteiros para linhas de unsigned char */
                  if(!char_mat) {
                    printf("\nErro na alocacao de ponteiros de linhas \
                            na matriz %dx%d \n",linha,coluna);
                    exit(1);
                   }
                  for(i = 0; i < linha; i++) {
        char_mat[i] = (unsigned char *)calloc(coluna,sizeof(unsigned char));
                           /*aloca ponteiros para colunas de unsigned char */
                    if(!char_mat[i]) {
                            printf("\nErro na alocacao de ponteiros de \
                              colunas em %d na matrix %dx%d \n",i,linha,coluna);
                            exit(1);
                     }
                   }
                  AA->ptr_mat = (char **)char_mat; /* ptr_mat aponta para */
                  break;                           /* a area de dados de AA */
                 }
                case sizeof(short): {   /* matriz de inteiros */
                  short **int_mat;
                  int_mat = (short **)calloc(linha,sizeof(short *));
                            /*Aloca ponteiros para linhas de int_mat */
                  if(!int_mat) {
                    printf("\nErro na alocacao de ponteiros de linhas \
                            na matriz %dx%d \n",linha,coluna);
                    exit(1);
                   }
                  for(i = 0; i < linha; i++) {
                    int_mat[i] = (short *)calloc(coluna,sizeof(short));
                             /*aloca ponteiros para colunas de int_mat */
                    if(!int_mat[i]) {
                            printf("\nErro na alocacao de ponteiros de \
                              colunas em %d na matrix %dx%d \n",i,linha,coluna);
                            exit(1);
                     }
                   }
                  AA->ptr_mat = (char **)int_mat; /* ptr_mat aponta para */
                  break;                          /* a area de dados de AA */
                 }
                case sizeof(float): {  /* matriz de floates */
                  float **float_mat;
                  float_mat = (float **)calloc(linha,sizeof(float *));
                            /*Aloca ponteiros para linhas de float_mat */
                  if(!float_mat) {
                    printf("\nErro na alocacao de ponteiros de linhas \
                            na matriz %dx%d \n",linha,coluna);
                    exit(1);
                   }
                  for(i = 0; i < linha; i++) {
                    float_mat[i] = (float *)calloc(coluna,sizeof(float));
                             /*aloca ponteiros para colunas de float_mat */
                    if(!float_mat[i]) {
                            printf("\nErro na alocacao de ponteiros de \
                             colunas em %d na matrix %d x %d \n",i,linha,coluna);
                            exit(1);
                     }
                   }
                  AA->ptr_mat = (char **)float_mat; /* ptr_mat aponta para */
                  break;                            /* a area de dados de AA */
                 }
                case sizeof(double): {  /* matriz de doubles */
                  double **double_mat;
                  double_mat = (double **)calloc(linha,sizeof(double *));
                            /*Aloca ponteiros para linhas de double_mat */
                  if(!double_mat) {
                    printf("\nErro na alocacao de ponteiros de linhas \
                            na matriz %dx%d \n",linha,coluna);
                    exit(1);
```

```
                    }
                    for(i = 0; i < linha; i++) {
                      double_mat[i] = (double *)calloc(coluna,sizeof(double));
                                /*aloca ponteiros para colunas de double_mat */
                      if(!double_mat[i]) {
                              printf("\nErro na alocacao de ponteiros de \
                                colunas em %d na matrix %dx%d \n",i,linha,coluna);
                              exit(1);
                      }
                    }
                    AA->ptr_mat = (char **)double_mat; /* ptr_mat aponta para */
                    break;                              /* a area de dados de AA */
                   }
                   default:
                    printf("\nErro na alocacao da matriz: tipo nao \
                              suportado \n");
                    exit(1);
            }
           return(AA);
           }
/*************************************************************************
 *************************************************************************
libera_mat   libera a area da matriz alocada para dados, bem como
             ponteiros e estruturas mapa MATRIX. Emite mensagem de
             Erro e provoca exit(1) caso ocorra uma passagem de estrutura
             impropria (ponteiro NULL ou matriz de tamanho zero ).

void libera_mat(MATRIX *AA)
*************************************************************************/
void libera_mat(MATRIX *AA)
           {
           int i;
           char **livre;

           /* checa se a estrutura eh valida */
           if(!AA || !AA->ptr_mat || !AA->linha || !AA->coluna) {
             printf("\nErro: a estrutura passada eh invalida \n");
             exit(1);
            }

           /* livre eh usado para liberar a area de dados */
           livre = AA->ptr_mat;

           /*libera cada linha de dados da matriz */
           for(i = 0; i < AA->linha; i++)
              free(livre[i]);

           /*libera cada ponteiro de linha da matriz */
              free((char *)livre);
              livre = NULL;   /* seta livre para nulo para erro */

              /*libera a estrutura MATRIX da matriz "AA" */
              free((char *)AA);
    }
/*************************************************************************
 *************************************************************************/
```

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "/home/galois/alexmr/cores/objects/kroutine/rgb_yuv/src/maniparq.h"


/*****************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*      Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*          Date: Nov 15, 1994
*      Verified:
*  Side Effects:
*  Modifications:
*****************************************************************/

/*---------------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|      Returns: TRUE (1) on success, FALSE (0) otherwise
|
|    Written By:
|         Date:
|  Modifications:
|
---------------------------------------------------------------*/
/*****************************************************************************
MANIPMAT.C  Rotinas de manipulacao de matrizes. Essas rotinas realizam
            operacoes do tipo soma, subtracao, multiplicacao ponto a
            ponto e produto direto de duas matrizes, extrai submatrizes,
            transpoen, inverte e extrai o determinante das mesmas.

*****************************************************************************/


/*****************************************************************************
escala()  multiplica uma matriz por um escalar de valor s. Retorna uma
          nova matriz escalada como um ponteiro para a nova estrutura. O
          escalar "s" eh uma constante double. Matrizes inteiras sao
          convertidas para float quando escaladas.

MATRIX *escalar(MATRIX *AA,double s)
```

```
*****************************************************************************/
MATRIX *escalar(MATRIX *AA,double s)
   {
    MATRIX *aloca_mat();
    MATRIX *BB;
    int lin_a, col_a;

    /* Atribui os valores de linha e coluna da matrix A */
    lin_a = AA->linha;
    col_a = AA->coluna;

   /* Escalar a matriz com os tipos tipos diferentes usando o switch() */
    switch(AA->tamanho) {
              case sizeof(short):  /* se AA eh int BB eh float */
                       BB = aloca_mat(lin_a,col_a,sizeof(float));
                       ESCALAR_MAT(AA,BB,*,s,lin_a,col_a,0,0,short,float)
                       break;
              case sizeof(float):  /* AA eh float BB eh float */
                       BB = aloca_mat(lin_a,col_a,sizeof(float));
                       ESCALAR_MAT(AA,BB,*,s,lin_a,col_a,0,0,float,float)
                       break;
              case sizeof(double):   /* AA eh double, BB eh double */
                       BB = aloca_mat(lin_a,col_a,sizeof(double));
                       ESCALAR_MAT(AA,BB,*,s,lin_a,col_a,0,0,double,double)
           }
          return(BB);

   }
/*****************************************************************************
 *****************************************************************************
sub_mat()  extrai uma sub_matriz de uma matriz maior dependendo da linha
           e coluna de offset que indicam a posicao de inicio da extracao.
           Retorna uma nova matriz menor como um ponteiro para a nova
           estrutura com o numero de linha e coluna entrado pelo usuario.

MATRIX *sub_mat(MATRIX *AA,int lin_off,int col_off,int lin,int col)
*****************************************************************************/
MATRIX *sub_mat(MATRIX *AA,int lin_off,int col_off,int lin,int col)
    {
           MATRIX *aloca_mat();
           MATRIX *BB;

    /* Checa se a sub_matriz excede o damanho da matriz maior */
          if((lin+lin_off) > AA->linha || (col+col_off) > AA->coluna) {
           printf("\nErro, a sub_matriz excede o tamanho da matriz maior\n");
           exit(1);
           }

    /* Aloca espaco para a sub_matriz de saida */
           BB = aloca_mat(lin,col,AA->tamanho);

    /* Extrai a sub_matriz usando os tipos diferentes com o switch() */
            switch(AA->tamanho) {
              case sizeof(short):  /* se AA eh int BB eh float */
                       ESCALAR_MAT(AA,BB,*,1,lin,col,lin_off,col_off,short,short)
                       break;
              case sizeof(float):  /* AA eh float BB eh float */
                       ESCALAR_MAT(AA,BB,*,1,lin,col,lin_off,col_off,float,float)
                       break;
              case sizeof(double):   /* AA eh double, BB eh double */
                       ESCALAR_MAT(AA,BB,*,1,lin,col,lin_off,col_off,double,double)
            }
           return(BB);

   }
/*****************************************************************************/
/*****************************************************************************
somar() funcao para operar duas matrizes, elemento por elemento,
           usando a macro ELEMENTO_MAT que realiza soma, subtracao e
           multiplicacao elemento a elemento das matrizes "A" com "B".
           O numero de linhas e colunas de ambas matrizes deve ser iguais.
           Retorna um ponteiro para a nova estrutura de matriz escalada "C".
           O tipo retornado depende dos tipos recebidos pela macro.
           Operacoes do tipo C = A+B, C = A-B, C = A.*B. O operador deve ser
           + (soma), - (subtracao) ou * (multiplicacao).
MATRIX *somar(MATRIX *AA,MATRIX *BB)
*****************************************************************************/
MATRIX *somar(MATRIX *AA,MATRIX *BB)
   {
           MATRIX *aloca_mat();
           MATRIX *CC;
           int lin_a, col_a;

    /* Checa se a matriz AA eh diferente da matriz BB */
          if(BB->linha !=  AA->linha || BB->coluna != AA->coluna) {
           printf("\nErro, a matriz BB deve ser do mesmo tamanho de AA \n");
           printf("\nAA eh %dx%d e BB eh %dx%d \n"
                                ,AA->linha,AA->coluna,BB->linha,BB->coluna);
           exit(1);
           }

    /* Atribui os valores de linha e coluna da matrix AA */
    lin_a = AA->linha;
    col_a = AA->coluna;

    /* Aloca espaco para a matriz CC. O tamanho de elemento sera o
          de maior tipo entre AA e BB */
          if(AA->tamanho > BB->tamanho)
```

```
                CC = aloca_mat(lin_a,col_a,AA->tamanho);
        else
                CC = aloca_mat(lin_a,col_a,BB->tamanho);

    /* Realiza a operacao de soma, subtracao ou multiplicacao ponto a
            ponto entre AA e BB usando os tipos diferentes com o switch() */
             switch(AA->tamanho) {
               case sizeof(short):  /* se AA eh int */
                        switch(BB->tamanho) {
                         case sizeof(short):  /* se BB eh int CC eh int */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,short,short,short)
                          break;
                         case sizeof(float):  /* se BB eh float CC eh float */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,short,float,float)
                          break;
                         case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,short,double,double)
                        }
                        break;
               case sizeof(float):  /* se AA eh float */
                        switch(BB->tamanho) {
                         case sizeof(short):  /* se BB eh int CC eh float  */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,float,short,float)
                          break;
                         case sizeof(float):  /* se BB eh float CC eh float */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,float,float,float)
                          break;
                         case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,float,double,double)
                        }
                        break;
               case sizeof(double):  /* se AA eh double */
                        switch(BB->tamanho) {
                         case sizeof(short):  /* se BB eh int CC eh double  */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,double,short,double)
                          break;
                         case sizeof(float):  /* se BB eh float CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,double,float,double)
                          break;
                         case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,+,double,double,double)
                        }
                        break;
             }
            return(CC);

    }
/*****************************************************************************
******************************************************************************
```

```
subtrair() funcao para operar duas matrizes, elemento por elemento,
            usando a macro ELEMENTO_MAT que realiza soma, subtracao e
            multiplicacao elemento a elemento das matrizes "A" com "B".
            O numero de linhas e colunas de ambas matrizes deve ser iguais.
            Retorna um ponteiro para a nova estrutura de matriz escalada "C".
            O tipo retornado depende dos tipos recebidos pela macro.
            Operacoes do tipo C = A+B, C = A-B, C = A.*B. O operador deve ser
            + (soma), - (subtracao) ou * (multiplicacao).
MATRIX *subtrair(MATRIX *AA,MATRIX *BB)
******************************************************************************/
MATRIX *subtrair(MATRIX *AA,MATRIX *BB)
    {
            MATRIX *aloca_mat();
            MATRIX *CC;
            int lin_a, col_a;

     /* Checa se a matriz AA eh diferente da matriz BB */
            if(BB->linha !=  AA->linha || BB->coluna != AA->coluna) {
             printf("\nErro, a matriz BB deve ser do mesmo tamanho de AA \n");
             printf("\nMatriz AA eh %dx%d e BB eh %dx%d \n"
                                ,AA->linha,AA->coluna,BB->linha,BB->coluna);
             exit(1);
            }

     /* Atribui os valores de linha e coluna da matrix AA */
      lin_a = AA->linha;
      col_a = AA->coluna;

     /* Aloca espaco para a matriz CC. O tamanho de elemento sera o
            de maior tipo entre AA e BB */
            if(AA->tamanho > BB->tamanho)
                    CC = aloca_mat(lin_a,col_a,AA->tamanho);
            else
                    CC = aloca_mat(lin_a,col_a,BB->tamanho);

    /* Realiza a operacao de soma, subtracao ou multiplicacao ponto a
            ponto entre AA e BB usando os tipos diferentes com o switch() */
             switch(AA->tamanho) {
               case sizeof(short):  /* se AA eh int */
                        switch(BB->tamanho) {
                         case sizeof(short):  /* se BB eh int CC eh int */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,short,short,short)
                          break;
                         case sizeof(float):  /* se BB eh float CC eh float */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,short,float,float)
                          break;
                         case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,short,double,double)
                        }
```

```
                    break;
                case sizeof(float):  /* se AA eh float */
                        switch(BB->tamanho) {
                         case sizeof(short):  /* se BB eh int CC eh float  */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,float,short,float)
                          break;
                         case sizeof(float):  /* se BB eh float CC eh float */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,float,float,float)
                          break;
                         case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,float,double,double)
                        }
                        break;
                case sizeof(double):  /* se AA eh double */
                        switch(BB->tamanho) {
                         case sizeof(short):  /* se BB eh int CC eh double  */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,double,short,double)
                          break;
                         case sizeof(float):  /* se BB eh float CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,double,float,double)
                          break;
                         case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,-,double,double,double)
                        }
                        break;
              }
             return(CC);

    }
/*************************************************************************
**************************************************************************
mult_piece() funcao para operar duas matrizes, elemento por elemento,
             usando a macro ELEMENTO_MAT que realiza soma, subtracao e
             multiplicacao elemento a elemento das matrizes "A" com "B".
             O numero de linhas e colunas de ambas matrizes deve ser iguais.
             Retorna um ponteiro para a nova estrutura de matriz escalada "C".
             O tipo retornado depende dos tipos recebidos pela macro.
             Operacoes do tipo C = A+B, C = A-B, C = A.*B. O operador deve ser
             + (soma), - (subtracao) ou * (multiplicacao).
MATRIX *mult_piece(MATRIX *AA,MATRIX *BB)
**************************************************************************/
MATRIX *mult_piece(MATRIX *AA,MATRIX *BB)
    {
             MATRIX *aloca_mat();
             MATRIX *CC;
             int lin_a, col_a;

     /* Checa se a matriz AA eh diferente da matriz BB */
            if(BB->linha !=  AA->linha || BB->coluna != AA->coluna) {
                printf("\nErro, em multiplicar(), a matriz BB deve ser do mesmo \
                                tamanho de AA \n");
                printf("\nMatriz AA eh %dx%d e matriz BB eh %dx%d \n"
                                ,AA->linha,AA->coluna,BB->linha,BB->coluna);
                exit(1);
                }

     /* Atribui os valores de linha e coluna da matrix AA */
      lin_a = AA->linha;
      col_a = AA->coluna;

     /* Aloca espaco para a matriz CC. O tamanho de elemento sera o
            de maior tipo entre AA e BB */
            if(AA->tamanho > BB->tamanho)
                        CC = aloca_mat(lin_a,col_a,AA->tamanho);
            else
                        CC = aloca_mat(lin_a,col_a,BB->tamanho);

    /* Realiza a operacao de soma, subtracao ou multiplicacao ponto a
           ponto entre AA e BB usando os tipos diferentes com o switch() */
            switch(AA->tamanho) {
              case sizeof(short):  /* se AA eh int */
               switch(BB->tamanho) {
                        case sizeof(short):  /* se BB eh int CC eh int */
                         ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,short,short,short)
                         break;
                        case sizeof(float):  /* se BB eh float CC eh float */
                         ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,short,float,float)
                         break;
                        case sizeof(double):   /* se BB eh double CC eh double */
                          ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,short,double,double)
                       }
                       break;
              case sizeof(float):  /* se AA eh float */
               switch(BB->tamanho) {
                        case sizeof(short):  /* se BB eh int CC eh float  */
                         ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,float,short,float)
                         break;
                        case sizeof(float):  /* se BB eh float CC eh float */
                         ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,float,float,float)
                         break;
                        case sizeof(double):   /* se BB eh double CC eh double */
                         ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,float,double,double)
                       }
                       break;
              case sizeof(double):  /* se AA eh double */
               switch(BB->tamanho) {
                        case sizeof(short):  /* se BB eh int CC eh double  */
                         ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,double,short,double)
```

```
                        break;
                       case sizeof(float):  /* se BB eh float CC eh double */
                        ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,double,float,double)
                        break;
                       case sizeof(double):   /* se BB eh double CC eh double */
                        ELEMENTO_MAT(AA,BB,CC,lin_a,col_a,*,double,double,double)
                     }
                     break;
                   }
                  return(CC);

   }
/*************************************************************************/
/*************************************************************************
multiplicar() funcao para realizar o produto de duas matrizes, "A" e "B"
                do tipo C = A*B usando a macro PRODUTO_MAT. O numero de linhas
                de "B" deve ser igual ao numero de colunas de "A". Retorna um
                ponteiro para a nova estrutura de matriz "C". O tipo
                retornado depende dos tipos recebidos pela macro. Emite erro
                e exit(1) se "A" e "B" nao estao corretas
MATRIX *multiplicar(MATRIX *AA,MATRIX *BB)
*************************************************************************/
MATRIX *multiplicar(MATRIX *AA,MATRIX *BB)
   {
           MATRIX *aloca_mat();
           MATRIX *CC;
           int lin_a, col_a, col_b;

    /* Checa se a matriz AA eh diferente da matriz BB */
           if(BB->linha !=  AA->coluna) {
            printf("\nErro, as linhas de BB deve ser iguas as colunas de AA \n");
            printf("\nMatriz AA eh %dx%d e BB eh %dx%d \n"
                              ,AA->linha,AA->coluna,BB->linha,BB->coluna);
            exit(1);
           }

    /* Checa de "AA" e "BB" sao matrizes validas */
           if(!AA->ptr_mat) {
             printf("\nErro, nenhuma matriz AA esta alocada para produto\n");
             exit(1);
           }

           if(!BB->ptr_mat) {
             printf("\nErro, nenhuma matriz BB esta alocada para produto\n");
             exit(1);
           }

    /* Atribui os valores de linha e coluna da matrix AA */
     lin_a = AA->linha;
     col_a = AA->coluna;
     col_b = BB->coluna;

    /* Aloca espaco para a matriz CC. O tamanho de elemento sera o
           de maior tipo entre AA e BB */
           if(AA->tamanho > BB->tamanho)
                   CC = aloca_mat(lin_a,col_b,AA->tamanho);
           else
                   CC = aloca_mat(lin_a,col_b,BB->tamanho);

    /* Realiza a operacao de produto entre AA e BB usando os tipos
           diferentes com o switch() */
            switch(AA->tamanho) {
             case sizeof(short):  /* se AA eh int */
              switch(BB->tamanho) {
                   case sizeof(short):  /* se BB eh int CC eh int */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,short,short,short)
                    break;
                   case sizeof(float):  /* se BB eh float CC eh float */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,short,float,float)
                    break;
                   case sizeof(double):   /* se BB eh double CC eh double */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,short,double,double)
              }
              break;
             case sizeof(float):  /* se AA eh float */
              switch(BB->tamanho) {
                   case sizeof(short):  /* se BB eh int CC eh float  */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,float,short,float)
                    break;
                   case sizeof(float):  /* se BB eh float CC eh float */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,float,float,float)
                    break;
                   case sizeof(double):   /* se BB eh double CC eh double */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,float,double,double)
              }
              break;
             case sizeof(double):  /* se AA eh double */
              switch(BB->tamanho) {
                   case sizeof(short):  /* se BB eh int CC eh double  */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,double,short,double)
                    break;
                   case sizeof(float):  /* se BB eh float CC eh double */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,double,float,double)
                    break;
                   case sizeof(double):   /* se BB eh double CC eh double */
                    PRODUTO_MAT(AA,BB,CC,lin_a,col_a,col_b,double,double,double)
              }
              break;
```

```
        }
        return(CC);

    }
/*************************************************************************/
/*************************************************************************
transposta() funcao para realizar a transposicao de uma matriz "A".
            Copia o resultado em uma nova matria At e retorna um ponteiro
            para a nova estrutura da matriz.
MATRIX *transposta(MATRIX *AA)
*************************************************************************/
MATRIX *transposta(MATRIX *AA)
        {
        MATRIX *aloca_mat();
        MATRIX *AAt;
        short **ai, **ait;
        float **af, **aft;
        double **ad, **adt;
        int i, j;

    /* Aloca espaco para a patriz AAt de saida */
        AAt = aloca_mat(AA->coluna,AA->linha,AA->tamanho);

    /* Realiza a transposicao de AA de varios tipos usando switch() */
        switch(AA->tamanho) {
                case sizeof(short):
                        ai = (short **)AA->ptr_mat;
                        ait = (short **)AAt->ptr_mat;
                        for(i = 0; i < AA->linha; i++)
                           for(j = 0; j < AA->coluna; j++)
                                ait[j][i] = ai[i][j];
                        break;
                case sizeof(float):
                        af = (float **)AA->ptr_mat;
                        aft = (float **)AAt->ptr_mat;
                        for(i = 0; i < AA->linha; i++)
                           for(j = 0; j < AA->coluna; j++)
                                aft[j][i] = af[i][j];
                        break;
                case sizeof(double):
                        ad = (double **)AA->ptr_mat;
                        adt = (double **)AAt->ptr_mat;
                        for(i = 0; i < AA->linha; i++)
                           for(j = 0; j < AA->coluna; j++)
                                adt[j][i] = ad[i][j];
                        break;
         }
        return(AAt);
    }

/*************************************************************************
*************************************************************************/
```

```
/*
 * Khoros: #RCSID#
 */

#if !defined(__lint) && !defined(__CODECENTER__)
static char rcsid[] = "Khoros: #RCSID#";
#endif

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Library Routine for coreslib
  >>>>
  >>>>     Private:
  >>>>     Static:
  >>>>     Public:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "internals.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>


/*************************************************************
*
*  Routine Name: function_name - short description of function
*
*      Purpose: This should be a complete description that anyone
*               could understand;  it should have acceptable grammar
*               and correct spelling.
*
*                           This is the header for a PUBLIC function
*
*       Input: argument1 - explanation
*              argument2 - explanation
*              argument3 - explanation
*
*      Output: argument4 - explanation
*              argument5 - explanation
*
*     Returns: TRUE (1) on success, FALSE (0) otherwise
*
*  Restrictions: Restrictions on data or input as applicable
*    Written By:
*          Date: Nov 15, 1994
*      Verified:
*  Side Effects:
*  Modifications:
*************************************************************/

/*-------------------------------------------------------------
|
|  Routine Name: foo - short description of foo()
|
|      Purpose: This should be a complete description that anyone
|               could understand;  it should have acceptable grammar
|               and correct spelling.
|
|                           This is the header for a PRIVATE or STATIC function
|
|       Input: argument1 - explanation
|              argument2 - explanation
|              argument3 - explanation
|
|      Output: argument4 - explanation
|              argument5 - explanation
|
|     Returns: TRUE (1) on success, FALSE (0) otherwise
|
|    Written By:
|          Date:
|  Modifications:
|
-------------------------------------------------------------*/
/* FILTROS.C - programas fontes das funcoes dos filtros FIR usados
   para a composicao e decomposicao dos sinais RGB->PALM->RGB e
   interpolacao para a composicao de um novo quadro com uma nova
   taxa (frequencia) de amostragem. */


/*************************************************************************
   fir_impar   Filtro FIR com N impar de amostras e operacao com float
   fir_par     Filtro FIR com N par de amostras e operacao com float

*************************************************************************/
/* ESTRUTURA DOS FILTROS FIR PARA AS ROTINAS */

typedef struct {
   unsigned int tamanho;   /* Numero de amostras do filtro */
```

```
   float *coef;          /* ponteiro para os coeficientes do filtro */
} FILTRO;
/*************************************************************************
fir_impar - Filtro para dados do tipo float. Solicita a estrutura para
            os coeficientes. Os dados de saida tem o mesmo tamanho dos
            dados de entrada.

void fir_impar(float *in,float *out,int comp,FILTRO *fir_imp)

 in          ponteiro float dos dados de entrada
 out         ponteiro float dos dados de saida
 comp        comprimento dos dados de entrada e saida
 FILTRO *fir_imp   ponteiro para a estrutura do filtro FIR N=impar
*************************************************************************/
/*************************************************************************/

void fir_impar(float *in,float *out,int comp,FILTRO *fir_imp)
{
 int i,j,hmed,NAF,extensao;
 float soma;
 float *h_inicio,*h_medio,*h_ptr,*data,*data1,*data_fim;
 float *data_ptr1,*data_ptr2,*data_med,*n1,*n2;

/* Determinacao dos dados iniciais */
   NAF = fir_imp->tamanho;  /* N = 33 */
   hmed = (NAF + 1)/2;      /* hmed = 17 */
   extensao = comp+NAF-1; /* ext. do vet. extentido 512+32=544 bytes */
   h_inicio = fir_imp->coef;  /* aponta para o inicio dos coeficientes */
   h_medio = h_inicio+hmed-1; /* aponta para a posicao 17 a partir do */
                              /*  inicio dos coeficientes */
   /* Aloca espaco para o vetor expandido data */
   data = (float *) calloc(extensao,sizeof(float));
   if(!data) {
           printf("\nErro da alocacao do vetor extendido data ");
           exit(1);
    }
   data1 = data;    /* aponta para a mesma posicao de memoria */

/* Aumentar o tamanho do vetor dos dados de entrada com N amostras para
  realizacao da filtragem com menas perdas no inicio e no fim dos dados
  filtrados */
  for(i = 1; i <= (hmed-1); i++)
     *data1++ = *in;   /* repete a primeira amosta no inicio */
  for(i = 0; i < comp; i++)
     *data1++ = *in++;  /* transfere os dados in para os dados data1 */
  in--;     /* recua uma posicao de mamoria  */
  for(i = 1; i <= (hmed-1); i++)
     *data1++ = *in;  /* repete a ultima amostra hmed-1 vezes no final */

/* Posiciona os ponteiro nos enderecos iniciais de dados */
   data_med = data+hmed-1;  /* (1+17-1=17 bytes a frente ) */
   data_fim = data+extensao-1;  /* (1+544-1=544 bytes a frente ) */
   data_ptr1 = data;  /* aponta para o inicio dos dados */
   data_ptr2 = data+NAF-1;  /* (1+33-1=33 bytes a frente de data ) */

/* Inicio da realizacao da convolucao dos dados com os coeficientes
  do filtro */
   for( i = 0; i < comp; i++) {  /* 0 ate comprimento do registro */
     /* posiciona os ponteiros no inicio dos dados envolvidos */
     h_ptr = h_inicio;
     n1 = data_ptr1;
     n2 = data_ptr2;
     /* multiplicacoes iniciais e acumulos dos resultados */
     soma = (*h_medio) * (*data_med);
     soma += (*h_ptr++) * ((*n2--)+(*n1++));
     for( j = 1; j < (hmed - 1); j++)  {  /* varia de 1 a 15 */
              soma += (*h_ptr++) * ((*n2--)+(*n1++));
            }
     *out++ = soma;
     /* Incrementa os ponteiros para o proximo ciclo de multiplicacao */
     data_ptr1++;
     data_med++;
     data_ptr2++;
     if(data_ptr2 > data_fim) {};
   } /* Fim da convolucao dos dados com o filtro */
}
/*************************************************************************/
/*************************************************************************/
/*************************************************************************
fir_par - Filtro atrasador de 1/2 amostra para composicao de um novo
          quadro de dados com uma nova taxa de amostragem, dados do tipo
          float. Solicita a estrutura para os coeficientes. Os dados de
          saida tem o mesmo tamanho dos dados de entrada porem com um
          atrazo de fase correspondente a 1/2 amostra.

void fir_par(float *in,float *out,int comp,FILTRO *fir_p)

 in          ponteiro float dos dados de entrada
 out         ponteiro float dos dados de saida
 comp        comprimento dos dados de entrada e saida
 FILTRO *fir_p   ponteiro para a estrutura do filtro FIR N=par
*************************************************************************/

void fir_par(float *in,float *out,int comp,FILTRO *fir_p)
{
 int i,j,hmed,NAF,extensao;
```

```
 float soma;
 float *h_inicio,*h_ptr,*data,*data_fim,*data1;
 float *data_ptr1,*data_ptr2,*n1,*n2;

/* Determinacao dos dados iniciais */
   NAF = fir_p->tamanho;   /* exemplo N = 32 */
   hmed = NAF/2;       /* hmed = 16 */
   extensao = comp+NAF-1; /* ext. do vet. extentido 512+31=543 bytes */
   h_inicio = fir_p->coef;  /* aponta para o inicio dos coeficientes */

   /* Aloca espaco para o vetor expandido data */
   data = (float *) calloc(extensao,sizeof(float));
   if(!data) {
            printf("\nErro da alocacao do vetor extendido data ");
            exit(1);
    }
   data1 = data;    /* aponta para a mesma posicao de memoria */

/* Aumentar o tamanho do vetor dos dados de entrada com N amostras para
  realizacao da filtragem com menas perdas no inicio e no fim dos dados
  filtrados */
   for(i = 0; i < (hmed-1); i++)   /* 15 amostras no inicio */
     *data1++ = *in;    /* repete a primeira amosta no inicio */
   for(i = 0; i < comp; i++)
     *data1++ = *in++;   /* transfere os dados in para os dados data1 */
   in--;       /* recua uma posicao de mamoria  */
   for(i = 0; i < hmed; i++)  /* 16 amostras no final do vetor data1 */
     *data1++ = *in;  /* repete a ultima amostra hmed vezes no final */


/* Posiciona os ponteiro nos enderecos iniciais de dados */
   data_fim = data+extensao-1;   /* (1+543-1=543 bytes a frente ) */
   data_ptr1 = data;   /* aponta para o inicio dos dados */
   data_ptr2 = data+NAF-1;  /* (1+32-1=32 bytes a frente de data ) */

/* Inicio da realizacao da convolucao dos dados com os coeficientes
  do filtro */
   for( i = 0; i < comp; i++) {  /* 0 ate comprimento do registro */
     /* posiciona os ponteiros no inicio dos dados envolvidos */
     h_ptr = h_inicio;
     n1 = data_ptr1;
     n2 = data_ptr2;
     /* multiplicacoes iniciais e acumulos dos resultados */
     soma = (*h_ptr++) * ((*n2--)+(*n1++));
     for( j = 1; j < hmed; j++)  {  /* varia da amostra 1 a amostra 16 */
              soma += (*h_ptr++) * ((*n2--)+(*n1++));
            }
     *out++ = soma;
     /* Incrementa os ponteiros para o proximo ciclo de multiplicacao */
     data_ptr1++;
     data_ptr2++;
     if(data_ptr2 > data_fim) {};
    } /* Fim da convolucao dos dados com o filtro */
}
/*************************************************************************/
```

Programas cabeçalhos

```
/*
 * Khoros: #RCSID#
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose:
  >>>>
  >>>>    Written By:
  >>>>
  >>>>         Date:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#ifndef _rgb_yuv_maniparq_h_
#define _rgb_yuv_maniparq_h_


            /*---------------------------------------*
            |     #include
            ---------------------------------------*/
#include "macromat.h"
#include "macrovec.h"

            /*---------------------------------------*
            |     #defines
            ---------------------------------------*/
/* DEFINICAO DOS TIPOS USADOS NA LEITURA/ESCRITA DOS ARQUIVOS */
#define CHAR              0   /* de -128 a 127  1 byte */
#define UNSIGNED_CHAR  1   /* de 0 a 255  1 byte */
#define INT           2   /* de -32768 a 32767  2 bytes */
#define UNSIGNED_INT   3   /* de 0 a 65535  2 bytes */
#define LONG_INT      4   /* de -2147483648 a 2147483647  4 bytes */
#define UNSIGNED_LONG  5   /* de 0 a 4294967295  4 bytes */
#define FLOAT             6
#define DOUBLE            7

            /*---------------------------------------*
            |     typedefs
            ---------------------------------------*/
```

```
            /*---------------------------------------*
            |     global variable declarations
            ---------------------------------------*/

            /*---------------------------------------*
            |     macros
            ---------------------------------------*/

            /*---------------------------------------*
            |     routine definitions
            ---------------------------------------*/


#endif /* _rgb_yuv_maniparq_h_ */
/* Don't add after this point */
```

```
/*
 * Khoros: #RCSID#
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>     Purpose:
  >>>>
  >>>>   Written By:
  >>>>
  >>>>        Date:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#ifndef _rgb_yuv_macrovec_h_
#define _rgb_yuv_macrovec_h_


        /*---------------------------------------*
        |     #include
        ---------------------------------------*/


        /*---------------------------------------*
        |     #defines
        ---------------------------------------*/


        /*---------------------------------------*
        |     typedefs
        ---------------------------------------*/


        /*---------------------------------------*
        |     global variable declarations
        ---------------------------------------*/


        /*---------------------------------------*
        |     macros
        ---------------------------------------*/
```

```
/*************************************************************************
MACROVEC.H  Macros prototipos para operacoes com vetor de dados. Deve
            ser dato ponteiro e tipo de dado para cada macro. Isso per-
            mite as macros trabalhar com qualquer tipo legal de dados
            do C ou C++ (int, float, double, unsigned int, char, unsig-
            ned char, etc.).
            Mais de um ponteiro pode apontar para o mesmo vetor de dese-
            jado. Assim, a operacao a=a*b pode ser realizada com
            MULT_VEC(a,b,a,10,int,int,int) onde 10 eh o numero de elemen-
            to de cada vetos.

Contem as seguintes macros:

ADSUBMULT_VEC(a,b,c,comp,oper,tipo_a,tipo_b,tipo_c)
PRODUTO_INTERNO_VEC(a,b,s,comp,tipo_a,tipo_b)
SOMATORIO_VEC(a,s,comp,tipo_a)
ESCALA_VEC(a,b,s,comp,tipo_a,tipo_b)
*************************************************************************
*************************************************************************
ADSUBMULT_VEC()  Realiza operacoes ponto a ponto entre dois vetores (a,b)
            de tipos especificados colocando o resultado no vetor c.
            As operacoes realizadas sao do tipo C=A+B, C=A-B ou
            C = A.*B.

            a       ponteiro para o vetor a
            b               ponteiro para o vetor b
            c               ponteiro para o vetor c
            comp    comprimento dos vetores
            oper    operacao  a ser realizada de +, - ou *
            tipo_a  tipo de dados do vetor a
            tipo_b  idem b
            tipo_c  idem c
*/

#define ADSUBMULT_VEC(a,b,c,comp,oper,tipo_a,tipo_b,tipo_c) { \
            tipo_a *_PTA = a; \
            tipo_b *_PTB = b; \
            tipo_c *_PTC = c; \
            int _IX; \
            for(_IX = 0; _IX < (comp); _IX++) \
               *_PTC++ = (tipo_c)((*_PTA++) oper (*_PTB++)); \
          }
/*************************************************************************
 *************************************************************************
PRODUTO_INTERNO_VEC() Realiza a soma do produto interno entre dois vetores
            (a,b) de tipos especificados colocando o resultado na variavel
            definida s.

             a      ponteiro para o vetor a
```

```
            b               ponteiro para o vetor b
            s               variavel usada para armazenar o resultado do
                            produto interno entre dois vetores (nao ponteiro)
            comp        comprimento dos vetores
            tipo_a      tipo de dados do vetor a
            tipo_b      idem b

WARNING: os vetores de dados de entrada nao sao colocados para o mesmo
         tipo da variavel s. Portanto, pelo menos um dos tipos dos vetores
         devem estar habilitado para representar o produto individual
         sem ocorrer overflow.
*/

#define PRODUTO_INTERNO_VEC(a,b,s,comp,tipo_a,tipo_b) {  \
          tipo_a *_PTA = a;  \
          tipo_b *_PTB = b;  \
          int _IX;  \
          s = (*_PTA++) * (*_PTB++);  \
          for(_IX = 1; _IX < (comp); _IX++)  \
             s += (*_PTA++) * (*_PTB++);  \
          }
/*****************************************************************************
 *****************************************************************************
SOMATORIO_VEC()  Realiza o somatorio do vetor "a" e armazena o resultado
                 na variavel "s" definida previamente.

            a       ponteiro para o vetor a
            s               variavel usada para armazenaar o resultado
                            (nao eh um ponteiro)
            comp        comprimento do vetor a
            tipo_a      tipo de dados do vetor a
*/

#define SOMATORIO_VEC(a,s,comp,tipo_a) { \
          tipo_a *_PTA = a; \
          int _IX; \
          s = (*_PTA++); \
          for(_IX = 1; _IX < (comp); _IX++) \
             s += (*_PTA++); \
          }
/*****************************************************************************
 *****************************************************************************
ESCALA_VEC() Realiza o produto de um escalar "s" pelo vetor "a" ou converte
             o tipo de dados do vetor "a" e armazena vetor escalado no
             vetor de saida "b".

            a       ponteiro para o vetor a de entrada
            b       ponteiro para o vetor b de saida
            s               variavel usada para escalar o vetor resultante
                            de saida (nao eh um ponteiro)
            comp        comprimento do vetor a
            tipo_a      tipo de dados do vetor a
            tipo_b      tipo de dados do vetor b
*/

#define ESCALA_VEC(a,b,s,comp,tipo_a,tipo_b) { \
          tipo_a *_PTA = (tipo_a *)a; \
          tipo_b *_PTB = (tipo_b *)b; \
          int _IX; \
          for(_IX = 0; _IX < (comp); _IX++) \
             *(_PTB)++ = (tipo_b)(s * (*(_PTA)++)); \
   }
/*****************************************************************************
 *****************************************************************************/

          /*-------------------------------------*
          |     routine definitions
          -------------------------------------*/


#endif /* _rgb_yuv_macrovec_h_ */
/* Don't add after this point */
```

```
/*
 * Khoros: #RCSID#
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
   >>>>
   >>>>      Purpose:
   >>>>
   >>>>    Written By:
   >>>>
   >>>>         Date:
   >>>>
   >>>> Modifications:
   >>>>
   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#ifndef _rgb_yuv_macromat_h_
#define _rgb_yuv_macromat_h_


            /*---------------------------------------*
            |     #include
            ---------------------------------------*/


            /*---------------------------------------*
            |     #defines
            ---------------------------------------*/


            /*---------------------------------------*
            |     typedefs
            ---------------------------------------*/


            /*---------------------------------------*
            |     global variable declarations
            ---------------------------------------*/


            /*---------------------------------------*
            |     macros
            ---------------------------------------*/
```

```
/*************************************************************
MACROMAT.H  macros prototipos para as funcoes ler_str(), ler_int()
            ler_float(), matrix() etc.
            Realizam operacoes com matrizes de dados. Deve ser en-
            viado ponteiros e tipos de dado para cada macro. Isso
            permite que seja realizado operacoes com os tipos legais
            de "C" (int, float ou double).
As macros incluidas sao:
MAX(a,b)            determina o maximo entre a e b
MIN(a,b)            determina o minimo entre a e b
ROUND(a)                    arredonda a para o inteiro mais proximo
ESCALAR_MAT(a,b,s,lin,col,lin_off,col_off,tipo_a,tipo_b) multipli-
                            ca a matriz a por um escalar s retornando
                            uma matriz b
ELEMENTO_MAT(a,b,c,lin_a,col_a,oper,tipo_a,tipo_b,tipo_c) realiza
                            operacoes entre duas matrizes, elemento a
                            elemento do tipo C=A+B, C=A-B ou C=A.*B.
                            podendo ser usado em operacoes de filtragem
                            em frequencia com o uso de mascara de fil-
                            tragem.
PRODUTO_MAT(a,b,c,lin_a,col_a,oper,tipo_a,tipo_b,tipo_c) realiza
                            operacoes de multiplicacao entre duas ma-
                            trizes do tipo C=A*B.
 *************************************************************/
#define MAX(a,b)  ( ( (a) > (b) ) ? (a) : (b) )
#define MIN(a,b)  ( ( (a) < (b) ) ? (a) : (b) )
#define ROUND(a)  ( ( (a) < 0 ) ? (int)((a)-0.5) : (int)((a)+0.5) )

/*************************************************************************
ESCALAR_MAT macro para operacoes de multiplicacao, divisao, soma ou
        subtracao elemento por elemento de uma matriz "A" com um
    escalar "s". Retorna um      ponteiro para a nova estrutura de
        matriz escalada "B". O tipo retornado depende dos tipos recebi-
        dos pela macro.

            a           ponteiro para a estrutura da matriz de entrada A
            b           ponteiro para a estrutura da matriz de saida B
            s           variavel fator de escala para ser multiplicado
            lin         numero de linhas da matriz B
            col         numero de culunas da matriz B
            lin_off     linha de offset da matriz A
            col_off     coluna de offset da matriz A
            tipo_a      tipo dos elementos da matriz A tipo legal do C ou C++
            tipo_b      tipo dos elementos da matriz B tipo legao do C ou C++

*/

#define ESCALAR_MAT(a,b,oper,s,lin,col,lin_off,col_off,tipo_a,tipo_b) { \
                    tipo_a **_AMX = (tipo_a **)a->ptr_mat; \
```

```
                tipo_b **_BMX = (tipo_b **)b->ptr_mat; \
                tipo_a *_PTA; \
                tipo_b *_PTB; \
        int _IX,_JX; \
                for(_IX = 0 ; _IX < lin ; _IX++) { \
          _PTB = _BMX[_IX]; \
                  _PTA = _AMX[_IX + lin_off] + col_off; \
                  for(_JX = 0 ; _JX < col ; _JX++) \
                        *_PTB++ = (tipo_b)(*_PTA++) oper (tipo_b)(s); \
        } \
      }

/*************************************************************************
 *************************************************************************
ELEMENTO_MAT macro para operacoes com matriz, elemento por elemento,
        para soma, subtracao e multiplicacao elemento a elemento das
        matrizes "A" com "B". O numero de linhas e colunas de ambas
        matrizes deve ser iguais. Retorna um ponteiro para a nova estrutura
        de matriz escalada "C". O tipo retornado depende dos tipos recebi-
        dos pela macro. Operacoes do tipo C = A+B, C = A-B, C = A.*B.

        a       ponteiro para a estrutura da matriz de entrada A
        b       ponteiro para a estrutura da matriz de saida B
        c       ponteiro para a estrutura da matriz de saida C
        lin_a   numero de linhas da matriz A
        col_a   numero de culunas da matriz A
        oper    operador sobre os elementos envolvidos. tipo legal do C
        tipo_a  tipo dos elementos da matriz A tipo legal do C ou C++
        tipo_b  tipo dos elementos da matriz B tipo legal do C ou C++
        tipo_c  tipo dos elementos da matriz C tipo legao do C ou C++
*/

#define ELEMENTO_MAT(a,b,c,lin_a,col_a,oper,tipo_a,tipo_b,tipo_c) { \
                tipo_a **_AMX = (tipo_a **)a->ptr_mat; \
                tipo_b **_BMX = (tipo_b **)b->ptr_mat; \
                tipo_c **_CMX = (tipo_c **)c->ptr_mat; \
                tipo_a *_PTA; \
                tipo_b *_PTB; \
                tipo_c *_PTC; \
        int _IX,_JX; \
                for(_IX = 0 ; _IX < lin_a ; _IX++) { \
          _PTC = _CMX[_IX]; \
          _PTB = _BMX[_IX]; \
          _PTA = _AMX[_IX]; \
                  for(_JX = 0 ; _JX < col_a ; _JX++) \
            *_PTC++ = (*_PTA++) oper (*_PTB++); \
        } \
      }
/*************************************************************************
 *************************************************************************
PRODUTO_MAT macro para multiplicacoes de matrizes, tipo C = A*B.
        Retorna um ponteiro para a nova estrutura de matriz "C".
        O tipo retornado depende dos tipos recebidos pela macro.
        Emite erro de exit(1) se o numero de colunas de "A" for dife-
        rente do numero de linhas de "B".

        a       ponteiro para a estrutura da matriz de entrada A
        b       ponteiro para a estrutura da matriz de saida B
        c       ponteiro para a estrutura da matriz de saida C
        lin_a   numero de linhas da matriz A
        col_a   numero de culunas da matriz A
        col_b   numero de culunas da matriz B
        tipo_a  tipo dos elementos da matriz A tipo legal do C ou C++
        tipo_b  tipo dos elementos da matriz B tipo legal do C ou C++
        tipo_c  tipo dos elementos da matriz C tipo legao do C ou C++
*/

#define PRODUTO_MAT(a,b,c,lin_a,col_a,col_b,tipo_a,tipo_b,tipo_c) { \
                tipo_a **_AMX = (tipo_a **)a->ptr_mat; \
                tipo_b **_BMX = (tipo_b **)b->ptr_mat; \
                tipo_c **_CMX = (tipo_c **)c->ptr_mat; \
                tipo_a *_PTA; \
                tipo_b *_PTB; \
                tipo_c *_PTC; \
        int _IX,_JX,_KX; \
                for(_IX = 0 ; _IX < lin_a ; _IX++) { \
                  _PTC = _CMX[_IX]; \
                  _PTB = _BMX[0]; \
                  for(_JX = 0 ; _JX < col_b ; _JX++) { \
                          _PTA = _AMX[_IX]; \
                          *_PTC = (*_PTA++) * (*_PTB++); \
                          for(_KX = 1 ; _KX < col_a ; _KX++) \
            *_PTC += (*_PTA++)* _BMX[_KX][_JX]; \
          _PTC++; \
        } \
      } \
    }
/*************************************************************************/
/* Macro para a composicao dos sinais PAL-M e NTSC  */
#define COMPOE(a,b,c,d,x,y,z,lin,col,tipo_a,tipo_d) { \
            tipo_a **_AMX = (tipo_a **)a->ptr_mat; \
            tipo_a **_BMX = (tipo_a **)b->ptr_mat; \
            tipo_a **_CMX = (tipo_a **)c->ptr_mat; \
            tipo_d **_DMX = (tipo_d **)d->ptr_mat; \
            tipo_a *_PTA; \
            tipo_a *_PTB; \
            tipo_a *_PTC; \
            tipo_d *_PTD; \
```

```
                float _XD; \
                int _IX, _IY; \
                for(_IX = 0; _IX < lin; _IX++) { \
                        _PTD = _DMX[_IX]; \
                        _PTA = _AMX[_IX]; \
                        _PTB = _BMX[_IX]; \
                        _PTC = _CMX[_IX]; \
                  for(_IY = 0; _IY < col; _IY++) { \
                        *_PTD++ = (tipo_d)(x*(*_PTA++) + y*(*_PTB++) + \
                        z*(*_PTC++)); \
                  } \
                } \
    }
/*****************************************************************************/
/*****************************************************************************/

        /*---------------------------------------*
        |     routine definitions
        ---------------------------------------*/


#endif /* _rgb_yuv_macromat_h_ */
/* Don't add after this point */
```

```
/*
 * Khoros: #RCSID#
 */

/*
 * Copyright (C) 1993 - 1996, Khoral Research, Inc., ("KRI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License or run klicense.
 */


/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  >>>>
  >>>>      Purpose:
  >>>>
  >>>>    Written By:
  >>>>
  >>>>         Date:
  >>>>
  >>>> Modifications:
  >>>>
  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#ifndef _coreslib_filtros_h_
#define _coreslib_filtros_h_


        /*---------------------------------------*
        |     #include
        ---------------------------------------*/


        /*---------------------------------------*
        |     #defines
        ---------------------------------------*/


        /*---------------------------------------*
        |     typedefs
        ---------------------------------------*/


        /*---------------------------------------*
        |     global variable declarations
        ---------------------------------------*/


        /*---------------------------------------*
        |     macros
                ---------------------------------------*/
/**********************************************************************
FILTROS.H  Macros e funcoes prototipos de filtros FIR com N impar e par
           para a realizacao de filtragem com filtros dN impar e
           interpolacao de dados com filtro de N par. Os dados deverao
           estar no formato float quando da realizacao da operacao de
           filtragem.

**********************************************************************
**********************************************************************/
/* Estrutura dos filtros FIR */
typedef struct {
           unsigned int tamanho;
           float *coef;
           } FILTRO;
/**********************************************************************/

/* Coeficientes h[n] do filtro passa baixa de 1.3 MHz com N = 33 (tamanho)
  amostras, KW1 = 4, T11 = 0.10965576 e T21 = 0.59674101. Onde KW1 e'
  o numero da ultima amostra de valor 1, T11 e o valor da segunda
  amostra de transicao da banda passante */

float fir1p3[33] = {
          1.0409500e-03,  2.5323667e-03,  2.9398184e-03, -5.7979062e-04,
    -7.3919585e-03, -1.1437272e-02, -5.1269569e-03,  1.1652690e-02,
          2.5890717e-02,  1.8818753e-02, -1.4969385e-02, -5.2843325e-02,
    -5.2345000e-02,  1.7073356e-02,  1.4230999e-01,  2.6466551e-01,
          3.1553921e-01,  2.6466551e-01,  1.4230999e-01,  1.7073356e-02,
    -5.2345000e-02, -5.2843325e-02, -1.4969385e-02,  1.8818753e-02,
          2.5890717e-02,  1.1652690e-02, -5.1269569e-03, -1.1437272e-02,
    -7.3919585e-03, -5.7979062e-04,  2.9398184e-03,  2.5323667e-03,
          1.0409500e-03 };

FILTRO F1p3 = { 33, fir1p3 };

/**********************************************************************/
/* Coeficientes h[n] do filtro passa baixa de 4.2 MHz com N = 33 (tamanho)
  amostras, KW2 = 13, T12 = 0.08137207 e T22 = 0.54712777. Onde KW2 e'
  o numero da ultima amostra de valor 1, T12 e' o valor da segunda
  amostra de transicao da banda passante */

float fir4p2[33] = {
    -1.4314110e-03,  3.4710585e-03, -3.3101747e-03,  3.0844682e-04,
          5.1527023e-03, -1.1648015e-02,  1.6958278e-02, -1.8549634e-02,
          1.4170983e-02, -2.4392172e-03, -1.6726740e-02,  4.1927911e-02,
    -7.0415743e-02,  9.8516189e-02, -1.2226783e-01,  1.3814653e-01,
          8.5627270e-01,  1.3814653e-01, -1.2226783e-01,  9.8516189e-02,
    -7.0415743e-02,  4.1927911e-02, -1.6726740e-02, -2.4392172e-03,
```

```
    1.4170983e-02, -1.8549634e-02,  1.6958278e-02, -1.1648015e-02,
    5.1527023e-03,  3.0844682e-04, -3.3101747e-03,  3.4710585e-03,
   -1.4314110e-03  };

FILTRO F4p2 = { 33, fir4p2 };

/*********************************************************************/
/* Coeficientes h[n] do filtro passa faixa de 3.58 MHz com N = 41 (tamanho)
   amostras, KS = 3, KD = 14, TV1 = 0.08393555, TV2 = 0.56226952,
   V1=V2=0, V3 = 0.08393555 e V4 = 0.56226952. Onde KS2 e' o numero
   da ultima amostra de valor 0 no inicio da faixa, KD e' o numero
   da ultima amostra de valor 1, TV12 e' o valor da primeira a mostra
   de transicao de subida e V1 e' o valor da ultima amostra de tran-
   sicao na descida da banda passante */

float fir3p58[41] = {
          1.7312869e-03, -2.8028048e-03, -4.2500282e-03,  1.9567783e-03,
   -3.2270241e-03,  7.3604113e-03,  1.6658904e-02, -7.0067658e-03,
          2.5726412e-03, -1.0991588e-02, -3.9829012e-02,  1.2049942e-02,
   -6.3927437e-04,  1.4171129e-02,  9.0229124e-02, -1.5732441e-02,
   -1.2816299e-03, -1.6512826e-02, -3.1272489e-01,  1.7233560e-02,
          5.0206876e-01,  1.7233560e-02, -3.1272489e-01, -1.6512826e-02,
   -1.2816299e-03, -1.5732441e-02,  9.0229124e-02,  1.4171129e-02,
   -6.3927437e-04,  1.2049942e-02, -3.9829012e-02, -1.0991588e-02,
          2.5726412e-03, -7.0067658e-03,  1.6658904e-02,  7.3604113e-03,
   -3.2270241e-03,  1.9567783e-03, -4.2500282e-03, -2.8028048e-03,
          1.7312869e-03  };

FILTRO F3p58 = { 41, fir3p58 };

/*********************************************************************/
/* Coeficientes h[n] do filtro interpolador (atrasador de 1/2 amostra)
   passa faixa com largura de faixa > 5.0 MHz, usado na reamostragem do
   sinal com N = 32 (tamanho) amostras, KW3 = 12, AT1 = 0.09323731 e
   AT2 = 0.56226952. Onde KW3 e' o numero da ultima amostra de valor 1
   AT1 e' o valor da primeira a mostra de transicao. */

float fir_interp[32] = {
          1.4763121e-03, -3.1698272e-03,  1.7393846e-03,  2.9877722e-03,
   -9.1543347e-03,  1.3413589e-02, -1.2209222e-02,  3.3607539e-03,
          1.2596169e-02, -3.1802788e-02,  4.7422703e-02, -5.0720885e-02,
          3.2023165e-02,  2.0946678e-02, -1.3989379e-01,  6.1098433e-01,
          6.1098433e-01, -1.3989379e-01,  2.0946678e-02,  3.2023165e-02,
   -5.0720885e-02,  4.7422703e-02, -3.1802788e-02, 1.2596169e-02,
          3.3607539e-03, -1.2209222e-02,  1.3413589e-02, -9.1543347e-03,
          2.9877722e-03,  1.7393846e-03, -3.1698272e-03,  1.4763121e-03,
          };

FILTRO FIR_interp = { 32, fir_interp };

/*********************************************************************
 *********************************************************************/

          /*-------------------------------------*
          |     routine definitions
          -------------------------------------*/
          /* programa FILTROS.C */
void fir_impar(float *,float *,int,FILTRO *);
void fir_par(float *,float *,int,FILTRO *);
/*********************************************************************/
/*********************************************************************/

#endif /* _coreslib_filtros_h_ */
/* Don't add after this point */
```