UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Daniel Duarte Abdala

Cyclops Personal – Uma Ferramenta para Gerenciamento e Visualização de Imagens Médicas no Padrão DICOM 3.0

TCC (Trabalho de Conclusão de Curso) submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Florianópolis, 5 de julho de 2002.

Agradecimentos

Ao Aldo, por permitir que eu trabalha-se em casa.
Aos LISHeiros pelo apoio
Ao CNPQ, pelo mesmo motivo que todo mundo agradece
Aos meus pais por acreditarem em mim e me bancarem
Aos meus irmãos Rachel, Déborah e Joel
A Aninha e a Penélope.

Banca Examinadora

_____

Pr. Dr. rer. nat. Aldo von Wangenheim (Orientador)

_____

Paulo Roberto Dellani (Banca)

_____

Herculano de Biasi (Banca)

_____

Pr. Renato Cislaghi (Coordenador)

1 Resumo

Com a criação e popularização dos exames baseados em imagens e sua posterior padronização pelo padrão DICOM (Digital Image Communication in Medicine), tornou-se viável um outro antigo sonho dos hospitais e clinicas, as PACS (Picture Archiving Communication System), que nada mais são do que integrar o parque computacional do hospital/clínica em uma rede e criar pontos de armazenamento de exames e entradas para consulta e análise dos mesmos.

Neste contexto, surgiu o projeto Cyclops, que além de tratar da criação e manutenção de PACS também se encarrega do desenvolvimento de aplicações para o auxílio ao diagnóstico.

Por requisições de desempenho e aceitação nacional, iniciou-se um esforço para que toda a implementação do Cyclops fosse portada para uma linguagem de maior desempenho. Neste contexto nasceu o projeto cyclops personal como um primeiro esforço que tanto implementa a base necessária para que o restante de projeto seja portado como funciona como um cliente pessoal para consulta, análise e diagnóstico de imagens médicas.

Palavras-chave: DICOM, imagens médicas, cliente de imagens médicas, Cyclops, Cyclops Personal

2 Abstract

With the creation and popularization of image based exams and your subsequent standardization for DICOM (Digital Image Communication in Medicine), became feasible another hospital and clinical dream, the PACS (Picture Archiving Communication System), that is just the hospital/clinic computational park interconnection in just one network, and create storage exams access points and inputs to query and analysis of then.

At this context, appeared the Cyclops Project, that beyond treat the creation and maintenance of PACS, to put it self of the application development to aid the exams diagnosis.

For performance and national acceptance requisitions, beginning a great effort to port all the Cyclops Project implementation to a programming language that has a better performance. Then, have borne the Cyclops Personal Project as a first effort that both implement the necessary base to remain project as function like a personal client for queries, analysis and diagnosis of medical images.

3 Sumário

4 Lista de Figuras

5 Lista de Exemplos

6 Lista de Tabelas

7 Introdução

O projeto Cyclops Personal, é parte integrante de um projeto maior chamado Cyclops, que por sua vez é um projeto de cooperação binacional entre o Brasil e a Alemanha e que visa o desenvolvimento de ferramentas de análise e diagnóstico de imagens médicas.

Dentro deste contexto, podemos classificar o cyclops personal de duas maneiras bem distintas.

Na primeira delas, o projeto pode ser visto como um cliente de imagens médicas no formato DICOM, possuindo alta performance e que pode facilmente ser executado em um computador de configurações modestas. Desta forma, um médico pode utiliza-lo em sua casa ou consultório para análises visuais ou auxiliadas por ferramentas básicas como, zoom, ajustes de contraste, e assim relatar seu laudo.

Mas o cyclops personal foi desenvolvido visando também objetivos maiores. Todas as aplicações desenvolvidas no âmbito do projeto cyclops possuem uma deficiência prática crassa. Elas foram desenvolvidas em linguagem de prototipação, ou melhor dizendo, em uma linguagem de performance modesta mas auto grau de abstração. Isso facilitou a criação de toda a tecnologia que compõe hoje o projeto cyclops, mas dificulta em muito a aceitação do software pela comunidade médica em nosso país, visto que para se executar de forma ao menos aceitável o programa, as configurações de hardware requeridas são bem altas.

Diante deste contexto, um novo esforço dentro do projeto cyclops está sendo feito no sentido de portar a tecnologia já desenvolvida para uma linguagem de maior performance. Mas para que tais aplicações funcionem, faz-se necessário que um background de software exista viabilizando tarefas genéricas tais como a capacidade do cliente se comunicar com servidores de imagens médicas ou dispositivos de captura de imagens (como um tomógrafo). Também e requerido como funcionalidades básicas a capacidade de carregar e interpretar de forma correta as imagens armazenadas no formato DICOM, manipulação das imagens como zoon e outras ferramentas e por último uma gama bem vasta de classes genéricas que se fazem necessárias para viabilizar serviços

que existiam na linguagem de prototipação utilizada e que não existem na nova linguagem adotada.

Desta forma, o cyclops personal foi projetado e desenvolvido de forma a cumprir tal papel e criar entradas bem definidas para todas as autoras aplicações a serem portadas.

Resumindo, poderíamos dizer que o cyclops personal é tanto uma ferramenta que pode ser utilizada por médicos no auxilio ao diagnostico médico como uma peça chave para que o projeto cyclops possa finalmente ser utilizado em larga escala em nosso país e não somente por nossos parceiros.

7.1 Historia do Projeto

O cyclops perssonal começou a ser desenvolvido em agosto de 2001 pelos bolsistas de ITI Daniel D. Abdala e Charles I. Wust do laboratório de Integração de software e hardware (LISHA), pertencente ‘a Universidade Federal de Santa Catarina.

Em dezembro de 2001 uma primeira versão estável do projeto foi finalizada contendo o kernel básico para interpretação de imagens médicas no formato DICOM 3.0, algumas ferramentas de manipulação de imagem e uma ferramenta de impressão.

De posse desta versão, o bolsista Charles I. Wust defendeu seu trabalho de conclusão de curso e deixou o projeto.

Em abril de 2002, uma nova versão foi lançada, versão esta contendo esquema de tratamento de exceções, kernel e parse de imagens expandido e corrigido, cliente de e-mail, uma primeira versão do mecanismo de comunicação e novas ferramentas de manipulação de imagens.

E o projeto continua a ser desenvolvido.

7.2 História do Projeto Cyclops

O projeto cyclops foi criado em 1992 como um projeto de pesquisa binacional e delongo prazo pelos professores Dr. Aldo von Wangenheim, da Universidade Federal

de Santa Catarina e pelo professor Dr. Michael M. Richter, da Universidade de Kaiserslautem.

Os objetivos do projeto são desenvolver e por em prática novos métodos, técnicas e ferramentas na área de análise de imagens médicas utilizando técnicas de inteligência artificial e visão computacional, além de criar uma estrutura bem definida de PACS que funciona em perfeito acordo com todo o projeto em si.. Com esta proposta, foi iniciada em 1993 a cooperação com os parceiros médicos e industriais da Alemanha. No Brasil existem também parceiros, tais como a clínica DMI em São José-SC e o Hospital Universitário da UFSC.

Atualmente, o projeto se encontra em sua fase II, que tem por objetivo a cooperação para o desenvolvimento de soluções que possam ser efetivamente transferidas dentro de aplicações clínicas práticas no contexto de um consórcio internacional. O objetivo deste consórcio é atingir as metas do projeto através de cooperação entre os parceiros do Brasil e da Alemanha, cujas competências específicas complementem-se em áreas específicas.

O consorcio internacional de pesquisa e desenvolvimento é constituído por Universidades de ambos os países, parceiros industriais da área de software, parceiros médicos e empresas fabricantes de equipamentos radiológicos de ambos os países.

7.3 Motivação

Durante os últimos nove anos, o projeto cyclops produziu pesquisa de mais alta qualidade resultando em ótimos softwares para análise e diagnóstico baseado em imagens médicas. Porém, toda a pesquisa desenvolvida teve como seu coroamento um software desenvolvido em linguagem smalltalk, que embora auxilie deveram durante o processo de desenvolvimento devido as facilidades inerentes a mesma no que se refere a prototipação de aplicações, possui uma performance considerada modesta, devido ao fato da linguagem não gerar código objeto compilado mas sim interpretado. Também vale notar que durante os mesmos nove anos, a equipe do projeto observou uma certa relutância por parte da comunidade médica em utilizar os aplicativos desenvolvidos pelo projeto devido ao fato dos mesmos não serem executados da maneira usual como qualquer programa para windows, além do fato de que as requisições de hardware para

uma execução aceitável dos mesmos ser bem alta. Diante deste contexto, em 2001, um novo esforço surgiu dentro do projeto cyclops no sentido de se criar uma nova versão do projeto que possui-se os seguintes requisitos:

- Alta performance
- Baixas requisições de hardware
- Compatibilidade total com o ambiente windows
- Mínimo esforço no processo de migração

Neste sentido, o primeiro passo a ser dado era a criação do kernel do projeto, ou seja, o suporte de software para perfazer as operações comuns a todos os aplicativos do projeto, tal como interpretar e abrir imagens no padrão DICOM, perfazer a comunicação de dados entre o computador que executa a operação e os repositórios de imagens e/ou dispositivos de captura das mesmas, dentre outras.

Então, o projeto cyclops personal surgiu como um primeiro esforço no sentido de se viabilizar o processo de migração de todos o projeto cyclops.

7.4 Objetivos

Para a idealização e desenvolvimento deste projeto foram definidos alguns objetivos visando principalmente o desenvolvimento de um software que possui-se um bom grau de usabilidade, que requisitasse o mínimo possível de recursos de software e hardware para que o software se adaptasse da melhor maneira possível aos computadores existentes hoje em dia nas clinicas e hospitais e também que contemplasse todas as funcionalidades previamente idealizadas e definidas no que consideramos um bom cliente de imagens médicas no que diz respeito a manutenção e manuseio. Desta forma definimos como objetivos gerais do projeto os seguintes itens:

7.4.1 Objetivos Gerais

O objetivo deste trabalho é desenvolver o suporte de software necessário para que o restante do projeto cyclops possa ser migrado de maneira efetiva para o ambiente windows. Como um subproduto deste projeto, visualizou-se a criação de um cliente de imagens médicas no padrão DICOM que possa ser utilizado em hospitais e clínicas no auxilio ao diagnóstico médico baseado em imagens.

7.4.2 Objetivos Específicos

Para realizar a implementação deste software foram definidos os seguintes objetivos específicos:

a) Criar um software capaz de ler informações contidas em um arquivo DICOM
b) O software deve fornecer funcionalidades que auxiliem os médicos no exame das imagens, tais como ferramentas de edição gráfica, e impressão.
c) Criar uma estrutura expansível, de modo que novas funcionalidades possam ser facilmente adicionadas ao software.
d) Criar um suporte de software de modo a tornar possível a migração dos aplicativos que compõem o projeto cyclops para o ambiente windows.

8 Metodologia

Para atingir os objetivos específicos destes trabalhos foi utilizada a seguinte metodologia:

a) Conhecer o padrão DICOM em detalhes, estudando suas definições no "Draft DICOM Standard" e em outros documentos.
b) Modelar e implementar uma estrutura para leitura da informação contida nos arquivos em formato DICOM.
c) Modelar e implementar uma estrutura de organização das informações lidas de acordo com a modelagem do mundo real proposta no padrão DICOM.
d) Adicionar ao sistema um visualizador de imagens com algumas funcionalidades, tais como ferramentas de "zoom".
e) Adicionar outras funcionalidades extras, tais como um mailer de Imagens ou impressão de imagens.
f) Criar artefatos de software que disponibilizem serviços para as aplicações que serão posteriormente portadas e que não existem na linguagem alvo escolhida.

8.1 Período de aquisição de Informações

Durante os primeiros dois meses do inicio do projeto, os desenvolvedores do mesmo dedicaram-se a aprender e adquirir informações necessárias para a realização do mesmo. Nesta etapa, pesquisou-se muito sobre o padrão de imagens médicas DICOM e também sobre tecnologias de software que auxiliassem no desenvolvimento da parte de manipulação de imagens de maneira concisa e eficiente. Entenda-se por tais tecnologias boa parte da API gráfica do Windows (GDI – Graphics Dependent Interface).

Neste Período também foram desenvolvidos alguns protótipos de partes do projeto, tais como a parte do parse de arquivos para extração das informações médicas e interpretação da representação das informações gráficas (imagens).

Embora este período de aquisição de informação tenha sido bem extenso e proveitoso, devido a complexidade do assunto no que diz respeito a concordância do aplicativo com o padrão DICOM (diga-se de passagem, que a ultima especificação do mesmo possui nada menos que 13 volumes, dos quais pelo menos 7 tiveram que ser detalhadamente analisados para que o resultado final fosse satisfatório), o período de estudo e aquisição de informações se entendeu até praticamente o último dia do projeto.

8.2 Escolha da Linguagem de Programação

Um dos maiores objetivos deste projeto é apresentar uma implementação mais eficiente que a aplicação "Dicom Editor", existente no âmbito do Projeto Cyclops.

Desta forma as preocupações que nortearam o início da idealização deste projeto recaíram sobre a escolha da linguagem de programação que deveria ser uma que possui-se alta performance na execução, e, diante deste requisito caíram por terra todas as linguagens que não são compiladas. O segundo requisito era que a linguagem fornecesse um bom grau de portabilidade, ou seja, que o programa pudesse ser escrito para windows e sem muito esforço, pudesse ser portado para outras plataformas, mais especificamente, o Linux. Embora a linguagem C++ possuísse alta performance, sua portabilidade deixa a desejar no quesito interface com o usuário. Esta linguagem, porém, teria que ser orientada a objetos, de modo a não perder uma das maiores qualidades que o DICOM Editor apresenta, que é uma boa modelagem do sistema, o que facilita a manutenção e expansão do mesmo.

Também era interessante que a linguagem tivesse bastante componentes visuais de fácil utilização, para facilitar o desenvolvimento de uma interface eficiente e agradável.

Desta forma, escolheu-se a linguagem Object-Pascal e as ferramentas Delhi e Kylix, pois, segundo pesquisas desenvolvidas no processo de seleção da linguagem percebeu-se que a aplicação poderia ser portada de uma plataforma para a outra sem grandes esforços.

8.3 Estrutura Geral da Aplicação

A estrutura geral da aplicação foi derivada diretamente a partir da aplicação ”DICOM Editor” já existente no projeto Cyclops.

Nesta estrutura, um usuário pode trabalhar com uma lista de imagens organizada a partir do modelo de informação do padrão DICOM. Assim, as imagens ficam divididas hierarquicamente em 4 níveis: paciente, estudo, série e imagem. O sistema exibe inicialmente a lista de todos os pacientes já obtidos. Selecionando um paciente serão mostrados todos os estudos pertencentes a este. As séries de um estudo também são visualizadas de modo similar.

Quando uma série está selecionada, o sistema realiza a visualização de miniaturas de imagens desta série. A partir deste passo é possível invocar funcionalidades do sistema especificas a uma série ou imagem

Esta estrutura hierárquica visualizada pela interface é mantida internamente por vários objetos que contêm as informações. Assim, foi necessário criar classes para cada um deles (pacientes, estudos, séries e imagens).

Uma vez que todos os requisitos do projeto estavam definidos, iniciou-se o processo de modelagem da aplicação. Para tal, utilizou-se ferramentas de modelagem em UML tal como o Rational Rose e também muitos diagramas de fluxo de informação melhor descritos de maneira textual. Toda a modelagem da mesma foi extensivamente utilizada no processo de desenvolvimento e também constantemente alterada durante as várias iterações no processo de desenvolvimento.

8.4 Metodologia de Desenvolvimento

De acordo com este processo de desenvolvimento, definiu-se uma estrutura geral para a aplicação, visando principalmente os seguintes quesitos:

- Modularidade
- Reusabilidade
- Portabilidade
- Performance

A modularidade era desejável principalmente porque durante a primeira etapa do desenvolvimento do projeto, dois programadores estavam trabalhando, ficando ao encargo de um principalmente o desenvolvimento do parse que interpretaria a informação DICOM contida nos arquivos .dcm e ao outro a representação da mesma em uma estrutura de dados concisa e que facilitasse sua utilização pelos vários outros módulos do projeto tal como o ImageViewer, o printer Tool, etc.

O quesito reusabilidade visava principalmente fornecer uma estrutura de objetos que pudesse ser facilmente entendida e reutilizada por outras aplicações do projeto cyclops que viessem a integrar o Cyclops Personal.

A portabilidade tinha como objetivo criar o mínimo possível de dependência a uma plataforma específica de modo a tornar simples o processo de transposição do sistema para um outro sistema operacional (no caso o linux).

Pro fim, a performance visava criar um aplicativo enxuto e que funcionasse a maior velocidade possível utilizando para isso o mínimo de possível de recursos da máquina hospedeira. Como o último requisito era considerado o mais importante e visto que o publico alvo do projeto (clinicas e hospitais) dificilmente aceitariam ter que mudar o parque de máquinas para contemplar as necessidades deste software e por fim visto que portabilidade e performance foram requisitos verificados contraditórios, negligenciou-se em parte a portabilidade visando a busca de uma maior performance.

8.5 [1]Documentação

Visto que esta aplicação era de suma importância para o projeto Cyclops e também que o mesmo seria continuado após o termino deste trabalho de conclusão de curso, não necessariamente pelos desenvolvedores originais, verificou-se a necessidade da criação de uma detalhada documentação do mesmo, visando principalmente documentar os processos críticos tais como a interpretação das informações no padrão DICOM, o processo de parse dos arquivos .dcm, os processos adotados para a manipulação eficiente de imagens no sistema operacional windows, em suma, todos os

[1] Toda a documentação pode ser encontrada no cd que acompanha o trabalho.

pontos onde se concentrou a pesquisa que validou tal aplicação como um projeto de conclusão de curso.

Para tal definiu-se que toda a aplicação deveria estar modelada em UML visto seu alto grau de aceitação e conhecimento pelas pessoas do meio da Computação. Também incluiu-se nesta documentação todas as descrições textuais dos processos complexos e relevantes do programa. Foi feita uma descrição detalhada de todas as classes do sistema especificando métodos, atributos e inter-relações intra e extra objetos.

Para finalizar a documentação, por todo o código fonte foram incluídos comentários explicando tecnicamente pontos chave da aplicação.

9 Conceituação Teórica

A aplicação desenvolvida neste trabalho tem como um de seus objetivos ler e decodificar imagens médicas no formato descrito pelo padrão DICOM 3.0. Desta maneira, é interessante fornecer detalhes mais aprofundados sobre este padrão para uma melhor compreensão dos capítulos seguintes deste relatório. Outro dos objetivos era que o programa manipulasse de forma eficiente as imagens médicas fornecendo ferramentas que fossem verdadeiramente úteis para os médicos, desta forma, também abordaremos neste capítulo informações técnicas sobre os procedimentos e tecnologias adotadas visando este fim.

9.1 Historia do Padrão DICOM

Algum tempo após o surgimento das primeiras imagens médicas em formato digital, o ACR e a NEMA perceberam a necessidade de padronização para PACS. Então, em 1983, foi formado um comitê conjunto para desenvolver um padrão para facilitar a conectividade entre diversos fabricantes de equipamentos de imagens médicas. Com este objetivo este comitê publicou em 1995 a versão 1.0 deste padrão com o documento chamado *ACR/NEMA Standards Publication No. 300-1985*. [CLUNIE, 2001]

Este documento sofreu várias revisões. Enfim, em 1982 foi lançada a versão 2.0 deste padrão, que foi denominada ACR/NEMA Standards Publication No. 300-1988. Esta versão realmente se tornou o padrão para comunicação e intercâmbio de imagens. Apesar de ter sido largamente utilizada, a versão do padrão sofria de muitas deficiências quanto a parte de comunicação em rede.

Então, o padrão sofreu uma revisão e reorganização radical e entre 1992 e 1993 foi publicada uma nova versão chamada ACR/NEMA Standards Publication PS3, que ficou mais conhecida pelo nome DICOM 3.0 (do inglês Digital Imaging and Communications in Medicine). [CLUNIE, 2001]

Após mais três anos de trabalho auxiliados por várias sugestões das áreas industrial e acadeêmica, o DICOM 3.0 foi dado por completo. Assim, esta versão acabou se tornando o padrão *de facto* para PACS, pois apresenta uma definição realmente

abrangente e robusta para comunicação e intercâmbio de imagens médicas. [DELLANI 2001]

A abrangência do padrão DICOM 3.0 em PACS não se restringe simplesmente a um protocolo de codificação dos dados da imagem e sua transmissão. Ele também define diversas classes de serviços, como armazenamento, recuperação, pesquisa e impressão de imagens, formatos utilizados no armazenamento das imagens em meios removíveis, processos de negociação de associações para a transmissão dos dados das imagens através de redes, etc. [DELLANI, 2001]

9.2 Documentação DICOM

O documento que define o padrão DICOM 3.0 encontra-se dividido em diversos capítulos. O modo em que estes foram divididos consegue dar uma boa noção da abrangência do padrão.

Os capítulos são (os de maior relevância para este projeto estão detalhados):

- **PS 3.1-2000,** *Introduction and Overvie*w: fornece uma introdução sobre os objetivos do padrão;

- **PS 3.2-2000,** *Conformanc*e: define fatores que devem ser observados para que uma aplicação possa ser considerada em conformidade com o padrão;

- **PS 3.3-2000** *Information Object Definition*s: define quais são os objetos de informação que são definidos pelo projeto e o modo como estes são constituídos. Contém uma extensa lista de todos os grupos e entidades de informação que o padrão define;

- **PS 3.4-2000** *Service Class Specification*s;

- **PS 3.5-2000** *Data Structures and Encoding*: define os métodos através dos quais os dados das estruturas de informações devem ser codificados;

- **PS 3.6-2000** *Data Dictionary*: é uma lista extensa com todos os "Tags" do padrão. Cada um destes "Tags" especifica cada um determinado atributo de um objeto de informação;

- **PS 3.7-2000** *Message Exchange*;

- **PS 3.8-2000** *Network Communication Support for Message Exchange*;

- **PS 3.9-2000** *Point to Point Communication Support for Message Exchange*;

- **PS 3.10-2000** *Media Storage and File Format for Media Exchange***:** especifica o formato de uma imagem DICOM em arquivo para armazenamento;

- **PS 3.11-2000** *Media Storage Application Profiles;*

- **PS 3.12-2000** *Media Formats and Physical Media for Media Interchange***;**

- **PS 3.13-2000** *Print Management Point-to-Point Communication Suppor***t;**

- **PS 3.14-2000** *Grayscale Standard Display Functio*n;

- **PS 3.15-2000** *Security Profiles.*

9.3 API-GDI do Windows

Num primeiro momento do projeto, mais precisamente quando a primeira versão do mesmo foi lançada, apenas utilizou-se os recursos disponíveis na linguagem de programação Delphi para gerenciar todo o processo de manipulação das imagens e também para a criação das ferramentas de manipulação de imagens. Considerando que a primeira versão nada mais era que um protótipo a ser validado para a criação efetiva de uma ferramenta, o projeto atendia em conformidade os objetivos. Mas em si, o sistema deixava muito a desejar. Por exemplo o processo de instanciação das imagens na memória era muito lento e também ferramentas como o ajuste window e a lupa de zoon apresentavam flicking acima do máximo aceitável.

Assim, fez-se necessário buscar uma solução para resolver os referidos problemas. Depois de um período de pesquisas, decidiu-se abrir mão das facilidades fornecidas pelo ambiente delphi no toante aos recursos de manipulação de imagens (perdendo assim a independência de plataforma) e adotar a estratégia de programar toda a manipulação de imagens utilizando diretamente a API-GDI do windows.

Programar utilizando diretamente a API do windows é tarefa bastante complexa, e para tal contou-se com a ajuda inestimável do livro de Charles Petzold, livro este que descreve de maneira detalhada e didática toda a API do windows. Os resultados obtidos com esta opção de desenvolvimento podem ser observados na seção referente aos resultados e testes de performance do projeto.

9.4 Estrutura de Informação DICOM

O padrão DICOM não define apenas métodos para comunicação de imagens. Ele também define toda uma modelagem de objetos de informação que vão além da própria imagem. Assim, a forma como as informações sobre pacientes, estudos e séries de imagens também tem uma definição de como devem ser transmitidas e armazenadas.

Assim sendo, cada imagem tem associada a s a si mesma um grande conjunto de informações. Como existem várias informações que existem de forma hierárquica é possível organizar as imagens da mesma maneira.

De acordo com o padrão DICOM, é possível organizar as informações no seguinte nível hierárquico:

a) **Paciente**: é uma pessoa que recebe ou está apto a receber tratamento médico. [PS 3.3, 2000]

b) **Estudo**: é uma coleção de séries de imagens médicas e outras informações logicamente relacionadas com o objetivo de diagnosticar um paciente. Um estudo está associado sempre a somente um paciente. [PS 3.3, 2000]

c) **Série**: séries são um conjunto de atributos utilizados para agrupar várias instancias de informação (comumente imagens) em diferentes conjuntos. Para agrupar várias instancias em uma série, todas as instâncias devem compartilhar o mesmo quadro de referência (*frame of reference*) e equipamento, caso tenham sido especificados, e ser da mesma modalidade. [PS 3.3, 2000]

d) **Imagem**: além da informação visual contém também outras informações relacionadas à mesma, tais como parâmetros da máquina de captura de imagens.

9.5 Formato Digital de Informações DICOM

O modo que o DICOM utiliza para codificar a informação é através de um dicionário de Elementos de Dados. Um elemento de dados especifica o valor de um determinado atributo de um objeto de informação.

A estrutura de um elemento de dados é composta dos seguintes campos:

a) **Tag**: É um identificador composto por um par ordenado de números: o Número de Grupo e o Número de Elemento. Estes números são cada um da ordem de 0000h a FFFFh (hexadecimal) [PS 3.5, 2000]. Os valores de tag são normalmente referenciados da seguinte forma (gggg, eeee), onde “gggg” e

"eeee" significam respectivamente os números de grupo e de elemento do tag em hexadecimal.

b) **Representação de Valor (VR, do inglês Value Representation)**: Define o tipo do dado codificado pelo elemento de dado (número, texto, etc). Este campo não é obrigatório.

c) **Comprimento do valor**: especifica o número de bytes utilizados para explicitar o valor do elemento de dados. [PS 3.5, 2000]

d) **Campo de Valor**: neste campo estão contidos os valores explicitados pelo elemento de dado

Desta maneira, no padrão DICOM, toda a informação sobre imagens é comunicada através de elementos de dados. O documento **PS 3.3-2000** *Information Object Definition*s define explicitamente quais elementos pertencem a cada diferente objeto de informação através do "Tag" deste elemento.

9.6 Considerações Sobre Performance

Como foi dito anteriormente, a performance da aplicação é um fator crítico visto que o volume de dados a ser manipulado pela aplicação é bastante grande. Apenas a titulo de exemplo, imagine que o software fosse usado para visualização, manipulação e diagnóstico de um exame tomográfico contendo apenas 10 imagens de resolução mediana. Para tal, o software deverá carregar na memória do computador cerca de 7 MB em imagens. Agora se considerarmos que dificilmente um médico trabalhará apenas com uma exame por vez, digamos, imagine 3 exames, o quantidade de memória pula para 21 MB. Acrescentemos a isso 5 MB ocupados pela aplicação em si. Logo chegamos a conclusão de que muita memória RAM será requisitada para o bom funcionamento do sistema. Para tornar mais desafiador o problema enfrentado pelo projeto, informamos que um exame tomográfico de alta resolução possui até 100 imagens.

Assim, fez-se necessário que desenvolvêssemos mecanismos para tornar possível a utilização do programa em computadores com recursos escassos, tal como os encontrados comumente em clinicas e hospitais. Algumas das técnicas adotadas foram

- Load on Demmand – ou seja, as imagens ficam armazenadas em arquivos temporários no disco rígido e só são carregadas quando for requisitado sua apresentação.
- Compactação – imagens não visíveis são compactadas
- Arraying and Bufferization – Dados necessários como os HU values que ocupam em geral o mesmo espaço que a imagem em memória são armazenados em arrays e buffers na memória estendida de modo a tornar a manipulação do window mais rápida.

A implementação de tais técnicas é discutida em maiores detalhes mais a frente neste trabalho.

9.7 Decodificação de Arquivos

Para realizar a construção da estrutura de informações DICOM é necessário realizar primeiramente a decodificação das informações contidas nos arquivos das imagens.

A unidade básica de informação em um arquivo DICOM é o elemento de dados. Os elementos de dados são estruturas que contêm o valor de cada um dos atributos de todos os objetos de informação relacionados a imagens, tais como pacientes, estudo, séries, entre outros.

Para realizar a decodificação de arquivos DICOM foi criado neste projeto um leitor destes arquivos. Este implementa funções para extrair de arquivos DICOM todos os elementos de dados contidos nele, e a partir deles gerar a informação necessária para a manipulação das imagens.

A estrutura como os vários campos de um elemento de dados são codificados é variável. Ela depende de qual “Sintaxe de Transferência” que a implementação do padrão estiver usando.

O padrão DICOM 3.0 define três tipos de sintaxe. Então, para que o sistema consiga extrair os elementos de dados de um arquivo DICOM, é necessário que este leitor implemente a leitura em cada uma destas sintaxes.

As sintaxes de transferência definidas pelo padrão DICOM são:

9.8 Sintaxe de Transferência "Implicit VR Little Endian"

Nesta sintaxe os elementos de dados devem observar as seguintes características:

O campo VR não deve existir nos elementos de dados codificados através desta sintaxe. Seu valor é relacionado a cada valor de Tag do elemento. Para saber qual é a representação de valor do campo basta consultar o documento **PS 3.6-2000** *Data Dictionary*.

As informações de cada campo devem ser codificadas utilizando a ordenação Little Endian, ou seja, o primeiro byte de um número contém a valor de menor significância, com os bytes seguintes em ordem **crescente** de significância.

Um elemento de dados codificado com esta sintaxe deve ter a seguinte configuração:

Núm. De Grupo	Núm. de Elemento	Comprimento do Elemento	Valor do Elemento
2 bytes	*2 bytes*	*4 bytes*	*[2]

Tabela 9-1 Formato de um Tag

O padrão DICOM define para cada sintaxe um Identificador Único (UID, do inglês Unique Identifier). O identificador universal desta sintaxe é "1.2.840.10008.1.2"

[2] *definido pelo valor contido no campo Comprimento do elemento*

9.9 Sintaxe de Transferência "Little Endian (VR Explicit)"

Nesta sintaxe os elementos de dados devem observar as seguintes características:

O campo VR deve existir obrigatoriamente em todos os elementos de dados. A partir de seu valor, é possível especificar qual é o tipo de dados contido dentro do elemento.

As informações de cada campo devem ser codificadas utilizando a ordenação Little Endian, ou seja, o primeiro byte de um número contém a valor de menor significância, com os bytes seguintes em ordem **crescente** de significância.

Um elemento de dados codificado com esta sintaxe deve ter uma das seguintes configurações, dependendo do tipo de representação de valor:

Núm. De Grupo	Núm. De Elemento	VR	Reservado	Comp. Do Elemento	Valor do Elemento
2 bytes	*2 bytes*	*2 bytes*	*2 bytes*	*4 bytes*	*[3]

Núm. De Grupo	Núm. De Elemento	VR	Comp. Do Elemento	Valor do Elemento
2 bytes	*2 bytes*	*2 bytes*	*2 bytes*	*[4]

O UID desta sintaxe de transferência é "1.2.840.10008.1.2.1"

[3] *definido pelo valor contido no campo Comprimento do elemento*

[4] *definido pelo valor contido no campo Comprimento do elemento*

9.10 Sintaxe de Transferência "Big Endian (Explicit VR)"

Esta sintaxe diferencia-se da anterior pelo fato de utilizar a notação Big Endian. Assim, nesta sintaxe o primeiro byte de um número contém a valor de maior significância, com os bytes seguintes em ordem **decrescente** de significância.

O UID desta sintaxe é "1.2.840.10008.1.2.2.".

Em todos os demais aspectos, ela é idêntica a sintaxe "Little Endian (Explicit VR)".

9.11 Estrutura de Arquivos DICOM

Em 1994 foi publicado o documento **PS 3.10-2000** *Media Storage and File Format for Media Exchange*. Um dos objetivos deste documento é especificar um modelo para o armazenamento de imagens e informações correlatas. Para tal, é definido um formato de arquivo que dá suporte ao encapsulamento de qualquer definição de objeto de informação.

Um arquivo DICOM é composto inicialmente de um conjunto de dados chamados de Meta Informação de Arquivo (*File Meta Information*). Este conjunto de dados é constituído primeiramente de um preâmbulo de 128 bytes, que caso a aplicação não os utilizem, devem ser preenchidos com o valor 00h. Em seguido existe um prefixo de 4 bytes com os caracteres "DICM" no formato do *Repertório de Caracteres ISO 8859 G0.* [PS 3.10, 2000].

Em seguida ao preâmbulo, dentro da Meta Informação do Arquivo, existe uma seqüência de elementos de dados, cuja codificação já foi esclarecida anteriormente. Todos estes elementos têm Tags com o número de grupo 0002h, e obrigatoriamente estão codificados com a sintaxe "Little Endian (Explicit VR)".

O primeiro elemento de dados deste conjunto define o comprimento em bytes do conjunto inteiro. Isto permite realizar uma leitura inicial de apenas os elementos do grupo com a Meta Informação do Arquivo.

Os dados contidos neste conjunto indicam diversas características do arquivo. Entre várias informações úteis para a comunicação das imagens, ele também define qual

é a sintaxe de transferência a ser utilizada pelo resto do arquivo. Esta informação é definida pelo elemento de dados “Transfer Syntax UID”, identificado pelo tag (0002h, 0010h).

Após a parte de Meta Informação do Arquivo, existe vários conjuntos de elementos de dados. Estes elementos contém cada um as informações de todos as entidades de informação relacionadas a imagem contida no arquivo. Assim, o arquivo mantém um registro de todos os dados sobre o paciente (por exemplo, nome, idade, sexo), estudo, série da imagem, entre outros.

O último grupo de elementos de dados (7FE0h) contêm as informações sobre os dados dos pixels. Ele é composto de dois elementos. O primeiro define o tamanho do valor do último, enquanto o último deixa seu campo de tamanho de valor indefinido (preenchido com o valor FFFFh). Este último elemento é onde ficam codificadas as informações do pixel do elemento em si.

9.12 Decodificação dos Pixels

A leitura dos arquivos permite a extração de todas as informações relacionadas a uma imagem. Porém, o modo como as próprias informações visuais da imagem são codificadas requer interpretação.

Os valores dos pixels da imagem estão contidos dentro elemento de dados identificado pelo tag (7FE0h, 0010h). Estes valores normalmente estão dispostos em uma matriz de pixels. Os valores dos pixels podem ser definidos por um campo de tamanho variável. Este tamanho é definido por um dos elementos de dados referentes à entidade de informação da imagem.

Porém, sua apresentação visual destes valores requer que eles sejam processados e devidamente convertidos para valores utilizados pelo computador, no caso deste projeto, a escala RGB.

Os valores contidos nos pixels podem ser definidos em várias escalas. No caso de imagens em escala de cinza, normalmente são utilizados valores na escala de Hounsfield.

As imagens, após convertidas, são gravadas temporariamente em disco rígido para evitar um grande consumo de memória. A cada sessão do sistema, cada imagem interpretada é gravada com um identificador único.

9.13 Imagens em unidade Hounsfield

A unidade de Hounsfield é uma medida da densidade relativa de uma estrutura em uma Tomografia Computadorizada. Ela foi batizada assim em homenagem a Sir Geoffrey Hounsfield, inventor da Tomografia Computadorizada. [CHENG, 2001]

Esta medida estipula que a densidade da água têm valor 0, enquanto a densidade do ar é tipicamente -1000. A seguir está uma tabela com exemplo de valores HU para algumas estruturas:

Estrutura	Valor em HU
Ar	-1000
Gordura	-100 a -40
Fluido	0 a 20
Tecidos leves	20 a 100
Osso	1000
Hemorragia intracranial aguda	55 a 75
Massa cinzenta	30 a 40

Tabela 9-2 Colelação de estruturas X HU values

Para a visualização de imagens codificadas nesta medida, é necessário realizar uma conversão destes valores para a escala RGB, comumente utilizada em sistemas gráficos. Para realizar esta operação, é necessário definir limites dentro dos valores em HU. Estes limites são normalmente chamados de “Window”.

Uma “Window” é definida dois valores em HU: Centro da “Window” (*Window Center*) e Comprimento da Window (*Window Width*). Estes dois valores definem um intervalo dentro da escala de HU da seguinte maneira: o número central deste intervalo é o Centro da Window, e o tamanho total deste intervalo é o Comprimento da Window. Assim:

$$\textbf{Window} = X \mid WC - \frac{WW}{2} \leq X \leq WC - \frac{WW}{2}$$

Sendo:

- WC = Centro da "Window" (*Window Center*)
- WW = Comprimento da "Window" (*Window Width*)

O seguinte figura demonstra melhor esta idéia, para uma Window com centro igual a 1000 e comprimento também igual 1000:

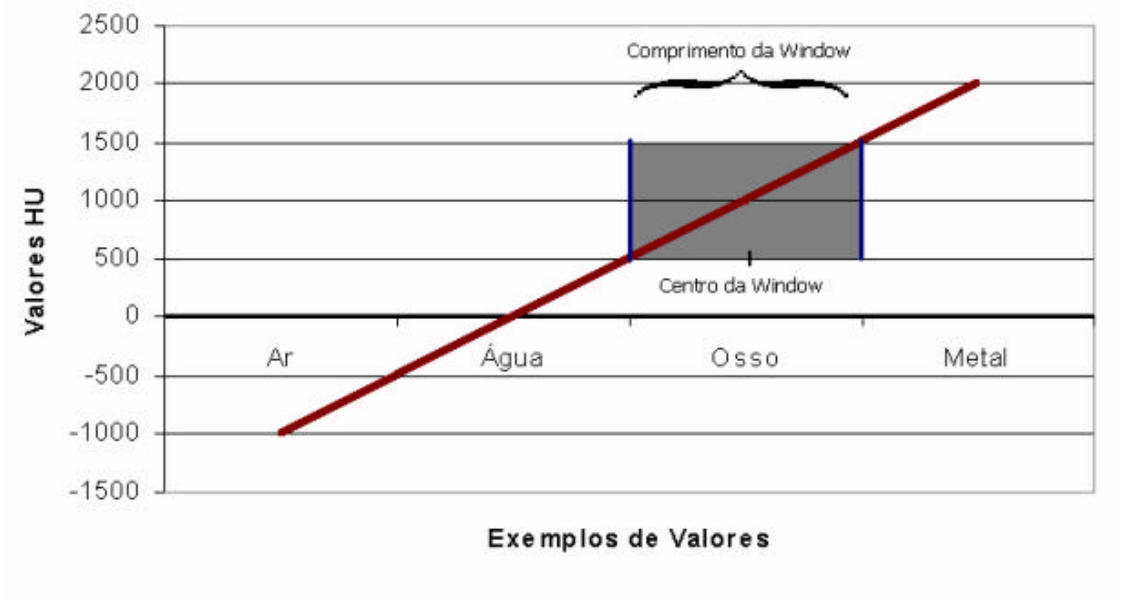


Figura 9.1 Exemplo de "Window" para valores em Unidades de Hounsfield

A conversão dos valores em HU para RGB pode ser feita fazendo com que valores inferiores ao intervalo da Window sejam definidos como preto (valor 0 em RGB), e valores maiores que o intervalo são mostrados como branco (valor 255). Os valores dentro do intervalo são definidos proporcionalmente entre 0 e 255.

Para realizar a conversão de um valor em HU para um intervalo da escala RGB, pode-se usar a seguinte fórmula:

$$RGB = \left(\frac{(HU - WC + \frac{WW}{2})}{WW} \right) * 256$$

Sendo:

- RGB = valor do pixel em escala de cinza de 0 a 255
- HU = Valor do pixel representado em HU
- WC = Centro da "Window" (*Window Center*)
- WW = Comprimento da "Window" (*Window Width*)

9.14 Imagens Coloridas

Para imagens com informações coloridas, como acontece em alguns tipos de ultra-sons, o formato de codificação é diferente.

Nestes tipos de imagens, não é necessário definir nenhum tipo de Window ou similar. Os valores representados na matriz de pixels existente no elemento de dados com a informação do pixel já vêm codificados na escala RGB.

O modo como os valores RGB são codificados nesta matriz é peculiar. Em tais imagens, a matriz de pixels vem triplicada. Na primeira matriz, estão dispostos todos os valores de vermelho da imagem. Na matriz seguinte, estão os valores de verde de cada pixel. A

última matriz contém os valores em vermelho do pixel.

9.15 Padrão de e-mail

Para que o cliente de e-mail fosse desenvolvido, fez-se necessário a leitura e entendimento das RFCs :

- rfc822
- rfc1225
- rfc1521
- rfc1725
- rfc821
- rfc2045

Estes documentos contém a descrição detalhada do modelo de e-mail corrente utilizado na internet, de seu esquema de codificação de imagens anexadas e dos mecanismos de POP3 e SMNP.

10 Estado da Arte

Como um dos objetivos gerais do projeto é criar um suporte de software para que as aplicações do projeto cyclops possam ser migradas para uma linguagem de maior performance que o Smalltalk, não existe, obviamente, nenhum software que desempenhe este papel.

10.1 Ferramentas Existentes

Porém, existem alguns programas que possuem a capacidade de decodificar imagens no padrão DICOM. Dentre eles podemos citar:

10.1.1 e-Film

O e-Film é o mais conceituado e conhecido cliente de imagens médicas existente na atualidade. Podemos dizer que ele é um software completo, possuindo ferramentas para comunicação em redes, ferramentas de auxilio a diagnóstico, filtros para alteração de contraste e ferramenta de impressão. Até o meio do ano 2001, este software era freeware, mas após esta data passou a ser cobrado. Outro ponto interessante é que o e-film possui sérios problemas com seu driver de impressão, inviabilizando algumas tarefas de impressão, quando o volume a ser impresso excede um certo limite.

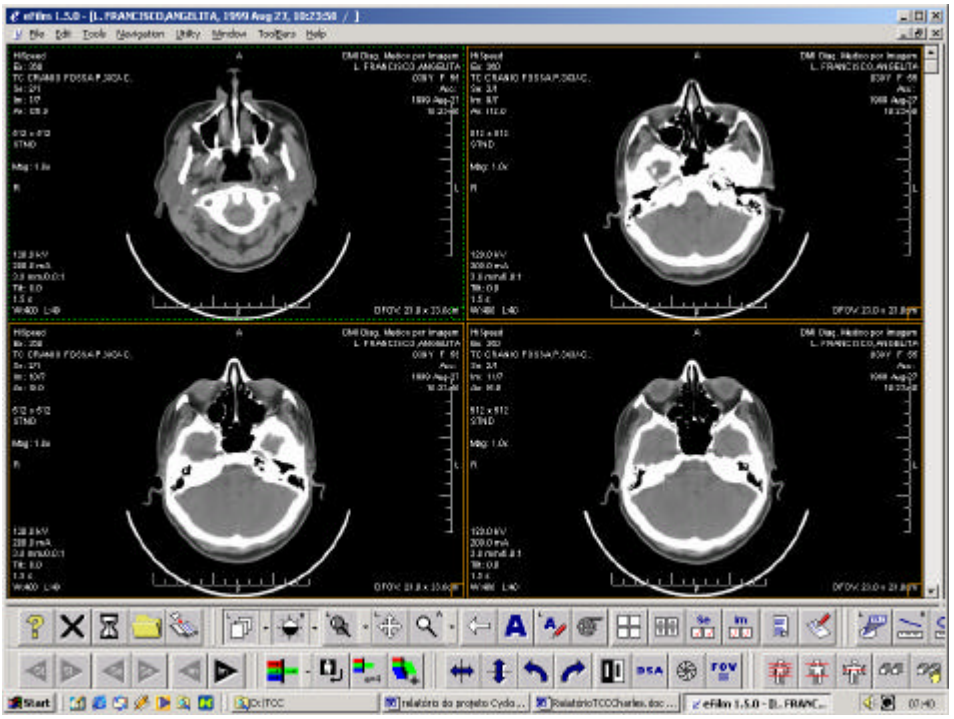


Figura 10.1 Janela do e-Film

10.1.2 ezDicom

O EzDicom é um software freeware que decodifica arquivos DICOM, inclusive imagens "multi-frame". Ele apresenta uma ótima performance. Porém, ele só trabalha com imagens DICOM separadas, e não dá pouco suporte para o trabalho com séries de imagens. O máximo que ele permite é a leitura da informação contida em cada um dos "Tags" que ela apresenta.

Ele também apresenta uma boa ferramenta para ajuste de Window (contraste e brilho da imagem), que é muito apreciada pelos médicos para auxiliar no diagnóstico de determinados problemas. O "zoom" na imagem também é permitido.

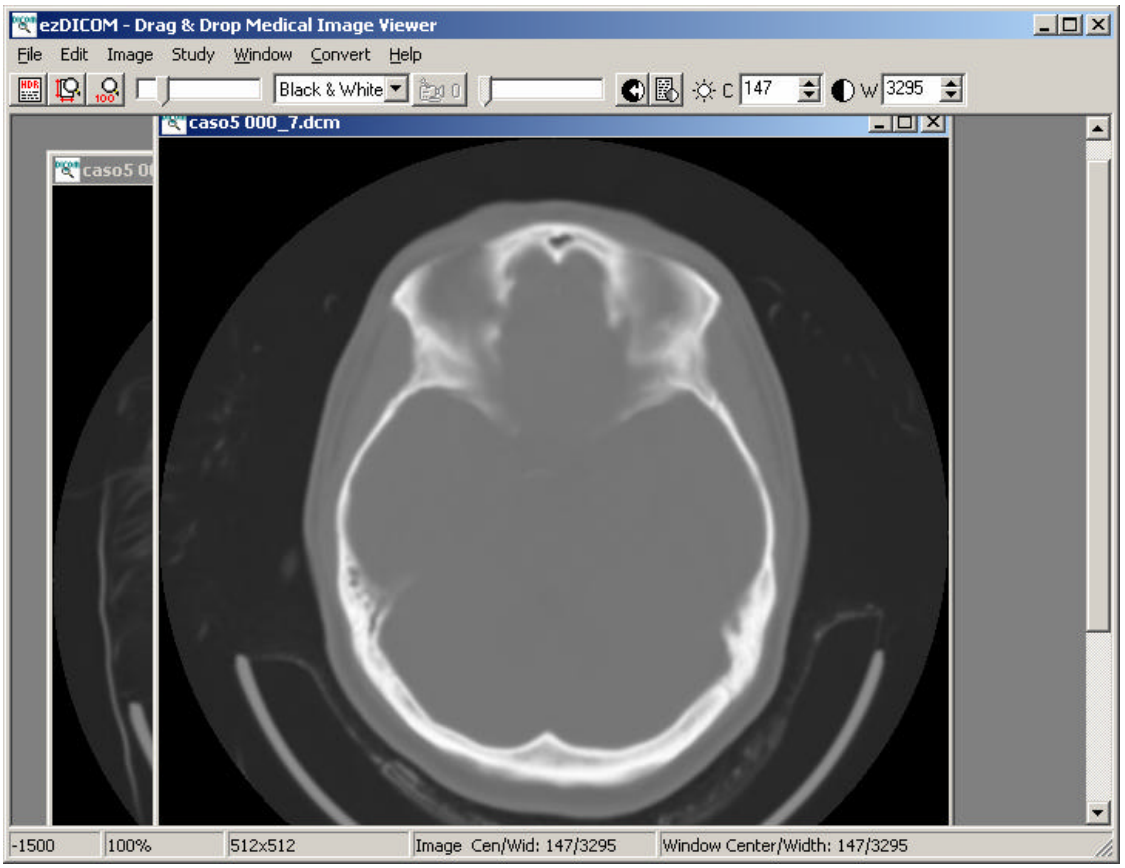


Figura 10.2 Janela do ezDICOM

10.1.3 LeadTools

O LEADTools é uma ferramenta comercial que também possui uma versão demo limitada que permite abrir e visualizar toda a informação contida em um arquivo DICOM. Interpreta corretamente a maioria dos tags do padrão, mas parece mais uma ferramenta de auxilio a desenvolvedores de software médico no padrão DICOM do que um cliente de auxílio a diagnóstico.

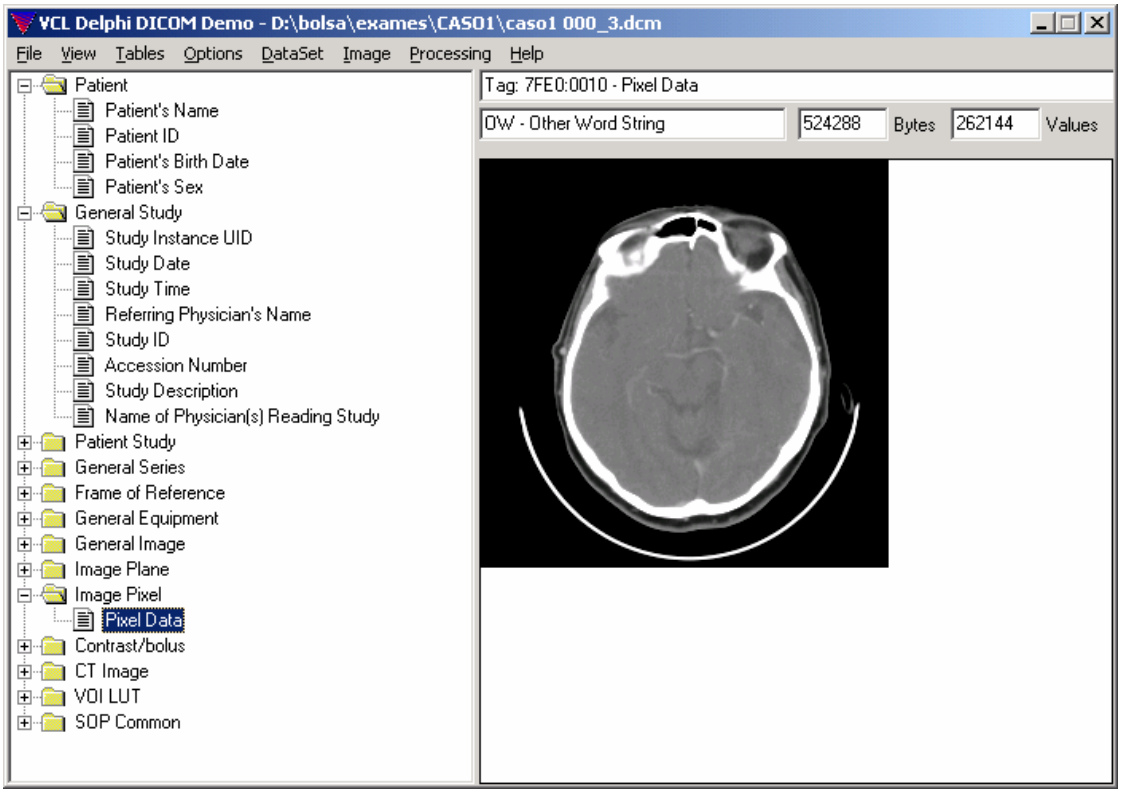


Figura 10.3 Janela do LeadTools

10.1.4 Osiris

Este programa foi por muito tempo considerado o melhor cliente para imagens médicas no padrão DICOM. Como podemos observar na figura, ele pede que especifiquemos alguns parâmetros da imagem. Este inconveniente tem feito que o mesmo caia em desuso embora ele possua um conjunto de ferramentas bem interessante e também comunicação via rede.

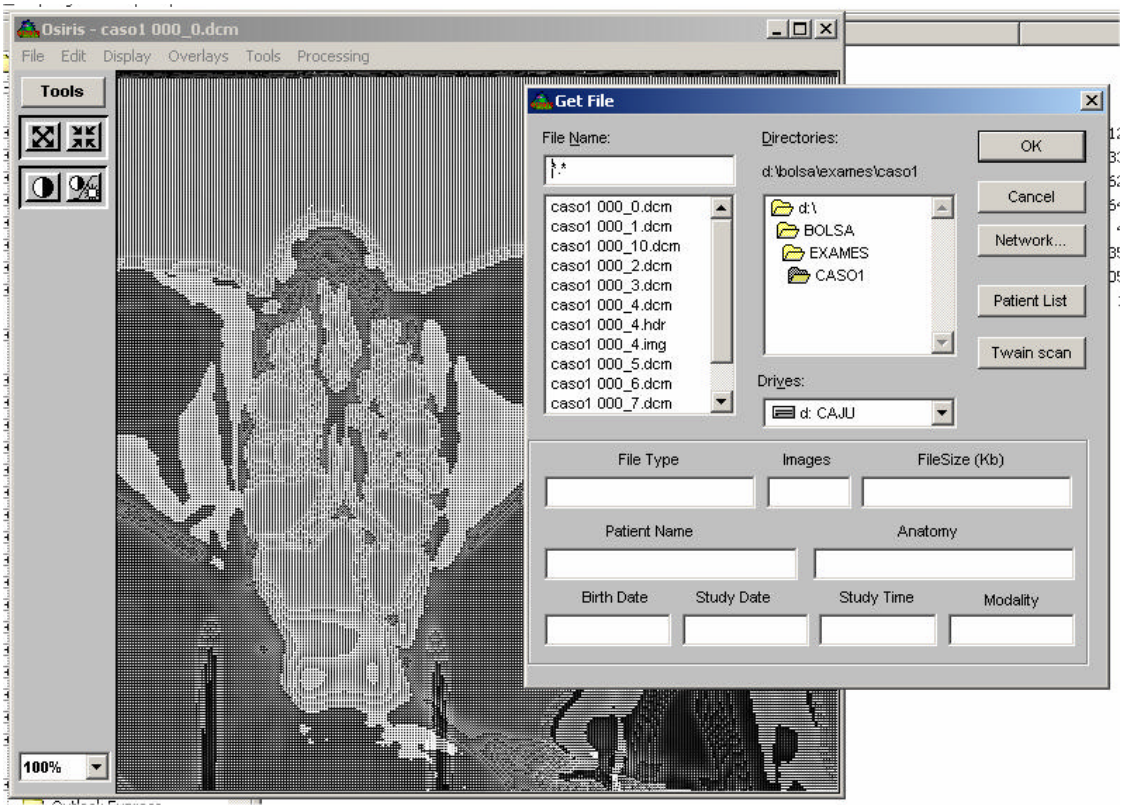


Figura 10.4 Janela do Osiris

10.1.5 Dicom Editor

Ferramenta desenvolvida no âmbito do projeto cyclops, possui uma boa interpretação do padrão DICOM, comunicação em redes, mas não possui ferramenta de impressão de imagens. Este software é a base do projeto cyclops servindo como trampolim para todas as outras aplicações desenvolvidas no projeto. Seu principal ponto negativo é sua performance, que só será aceitável se o computador que o executar for extremamente potente. Vale lembrar que o software foi desenvolvido em Smalltalk, o que significa que roda em pelo menos sete sistemas operacionais diferentes, entre eles estão o Windows, MacOS, UNIX, Linux, AIX, Solaris, etc.

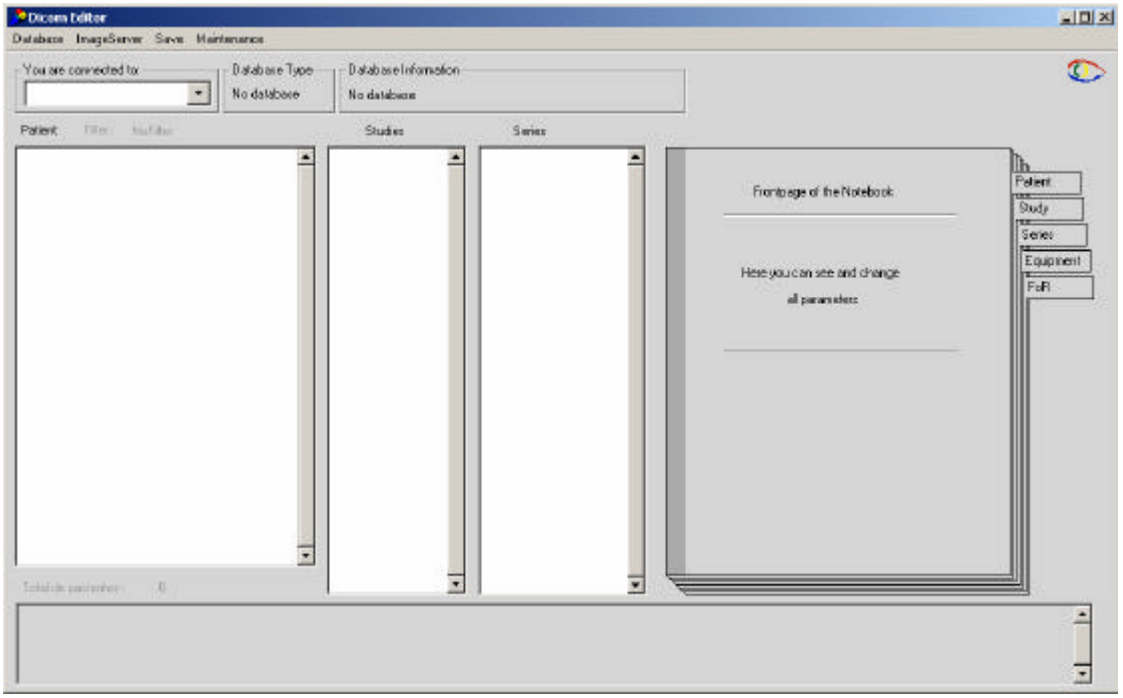


Figura 10.5 Janela do DICOM Editor

10.2 Requisitos do Mercado

Vale ressaltar que o mercado médico é extremamente exigente no que diz respeito a qualidade. Também devemos ressaltar que a velocidade de execução é importantíssima visto que a medicina em si exige dinamismo. Desta forma, após analisar todas as ferramentas existentes, observamos que praticamente todas pecava em um dos dois requisitos citados acima. Como exemplo podemos citar o ezDICOM que simplesmente não abre as imagens com o window em que as mesmas foram capturadas. Ao invés disso ele define o window considerado padrão. O e-Film por exemplo demora um tempo extremamente grande para imprimir as imagens, fato esse que dificilmente é tolerado num ambiente dinâmico de uma clínica. De posse destas informações, definimos um conjunto de requisitos a serem contemplados software.

Ainda outros requisitos foram levantados graças a cooperação dos parceiros médicos do projeto Cyclops. Dicas referentes a usabilidade e uma definição de

performance ótima, aceitável e indesejável para os mesmos. Tais dados podem ser vistos na seção referente a testes de performance e validação.

11 Características do Projeto

Tendo em vista os requisitos do projeto, o mesmo começou a ser desenvolvido em agosto de 2001. A seguir apresentamos algumas das funcionalidades criadas para cumprir os objetivos gerais e específicos.

11.1 DICOM Editor

Parte primordial do projeto, pois é a partir desta interface que se pode fazer praticamente tudo no programa. A organização das informações no esquema de pacientes, estudos, séries e imagens, está de acordo com o modelo do mundo real definido no draft do DICOM e foi em grande parte inspirada no DICOM editor oreviamente existente no projeto Cyclops.

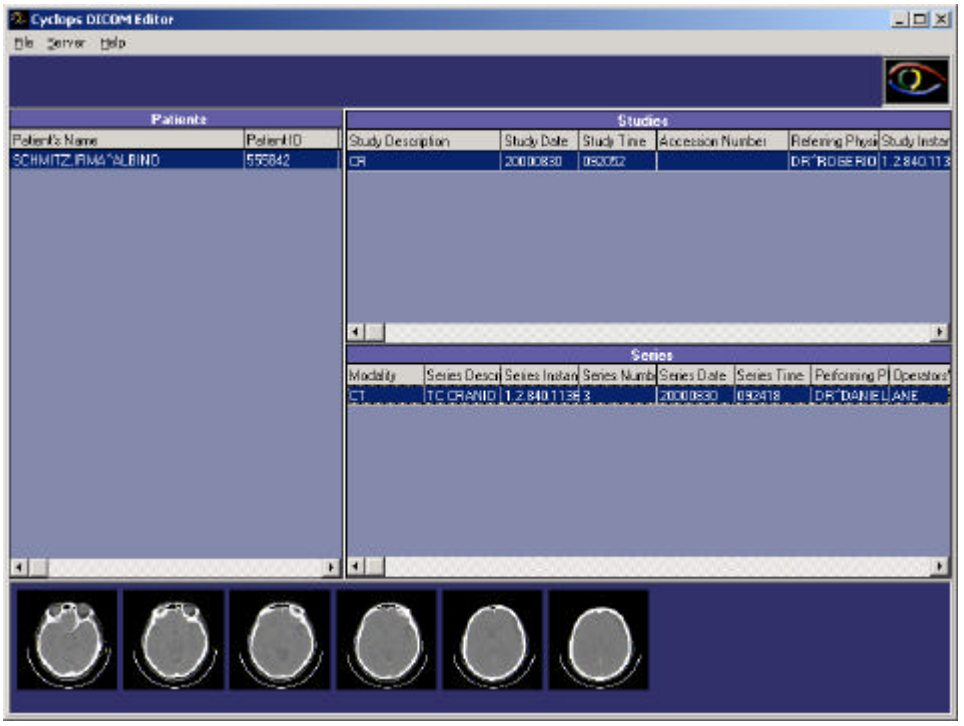


Figura 11.1 Dicom Editor main

11.2 Image Viewer

Ferramenta de auxilio a diagnóstico desenvolvida no sistema. Ela tem como principal objetivo permitir ao médico visualizar várias imagens ao mesmo tempo (geralmente da mesma série), e perfazer operações gráficas que possam auxilia-lo no diagnóstico. Dentre tais ferramentas podemos encontrar ferramentas de desenho, mensuração, ajuste de contraste, geração de imagens piloto, ajustes de zsoom e posicionamento das imagens na telça, rotação de imagens, etc.

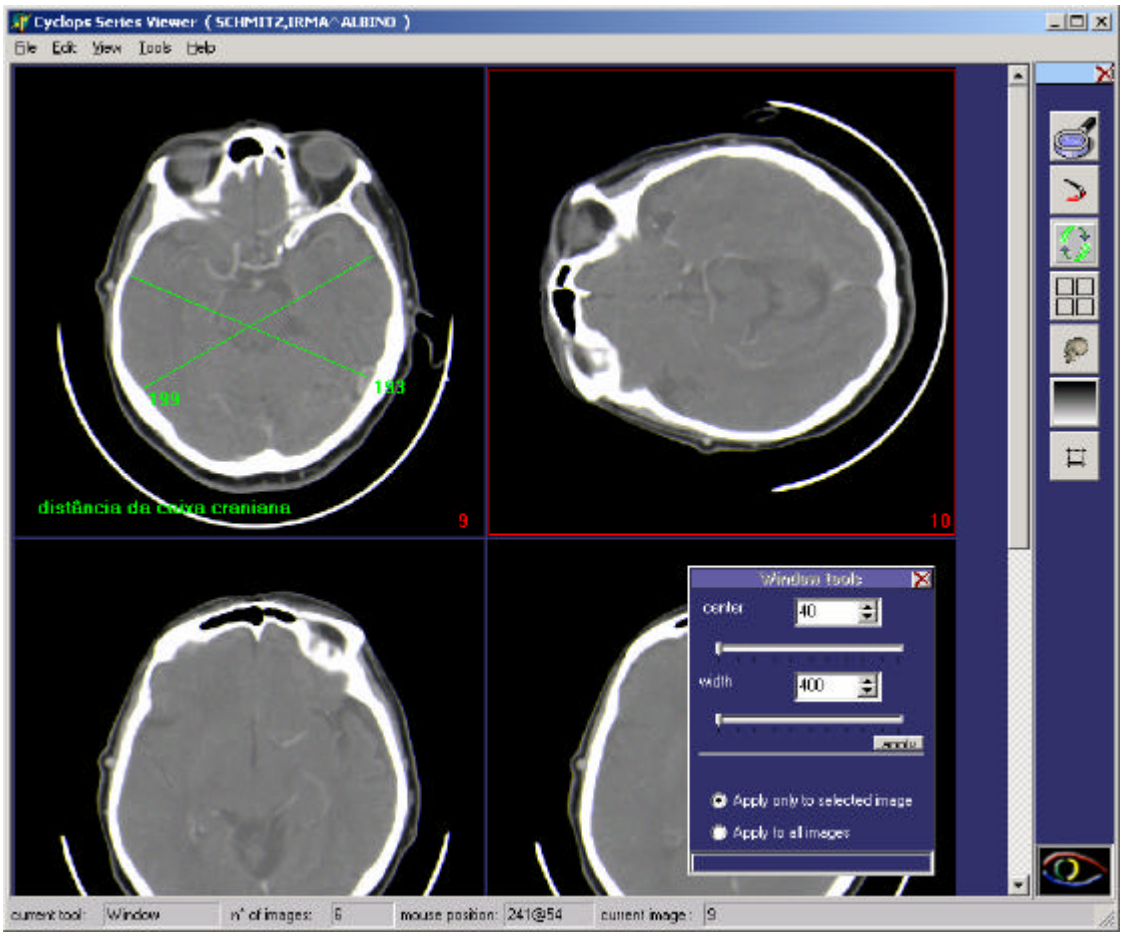


Figura 11.2 Image Viewer

11.3 Printer Tool

Um dos principias problemas encontrados no software e-film é o fato de que seu driver de impressão gera arquivos temporários de grande tamanho durante o processo de impressão. Em destes desenvolvidos na clinica DMI, onde existe uma impressora da XEROX específica para impressão de imagens médicas, constatou-se que para se imprimir uma folha contendo seis imagens de tomografia, uma arquivo de 30MB era gerado e o processo demorava quase 3 minutos. Diante deste quadro, a ferramenta de impressão do cyclops personal foi desenvolvida de modo a minimizar estes recursos, obtendo uma melhor performance que o software anteriormente mencionado.

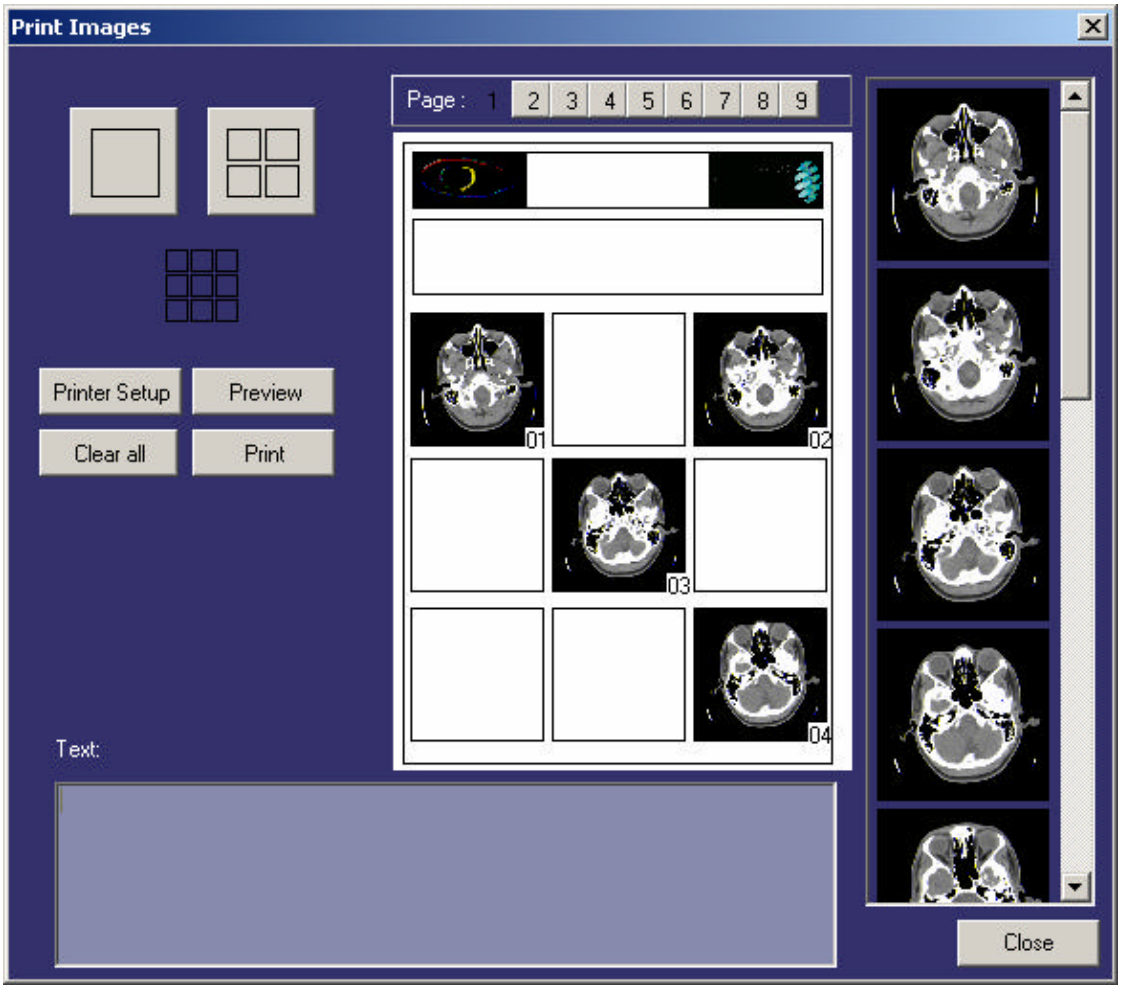


Figura 11.3 Printer Tool

11.4 Mailer

É muito comum no contexto de uma clinica de exames médicos que após ser efetuado um exame de captura de imagens como por exemplo uma tomografia, o radiologista responsável de o laudo e envie as imagens mais interessantes juntamente com o laudo para o médico que requisitou o exame. Isso acelera o processo de diagnóstico do médico requisitante, mas sem uma ferramenta especifica e integrada ao cliente de imagens DICOM, esta tarefa pode ser bastante cansativa. Diante deste quadro desenvolveu-se todo um novo cliente de e-mail que integrasse as facilidades do correio eletrônico as necessidades dos médicos.

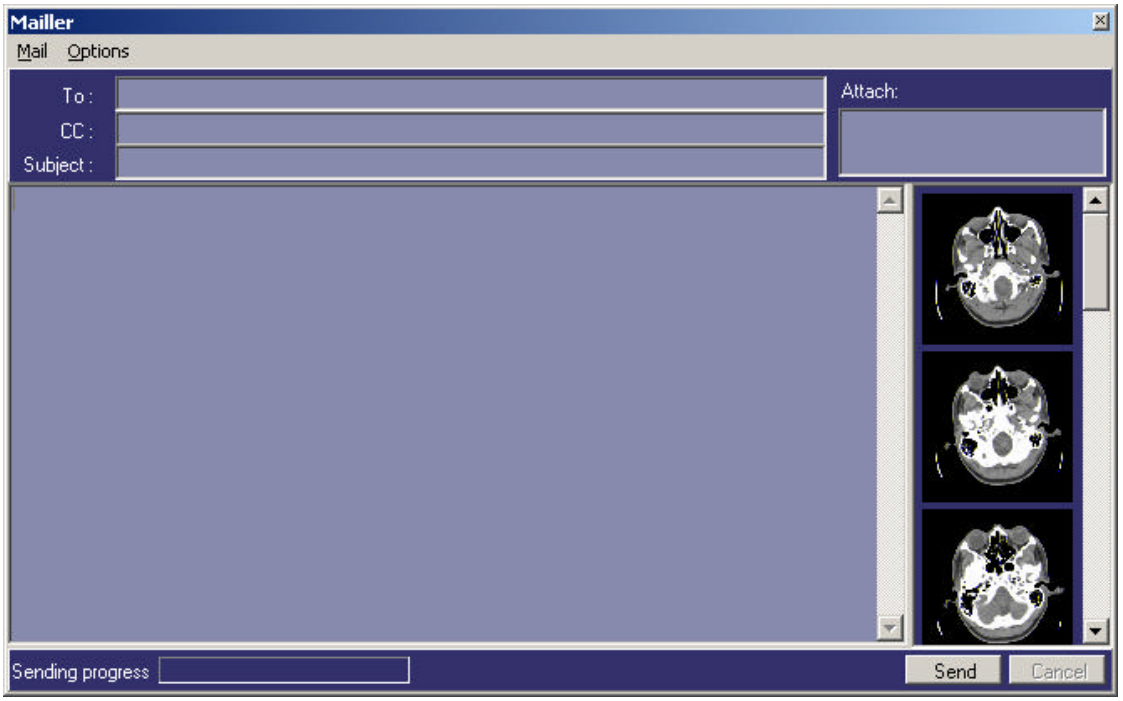


Figura 11.4 Mailer

11.5 Mini Viewer

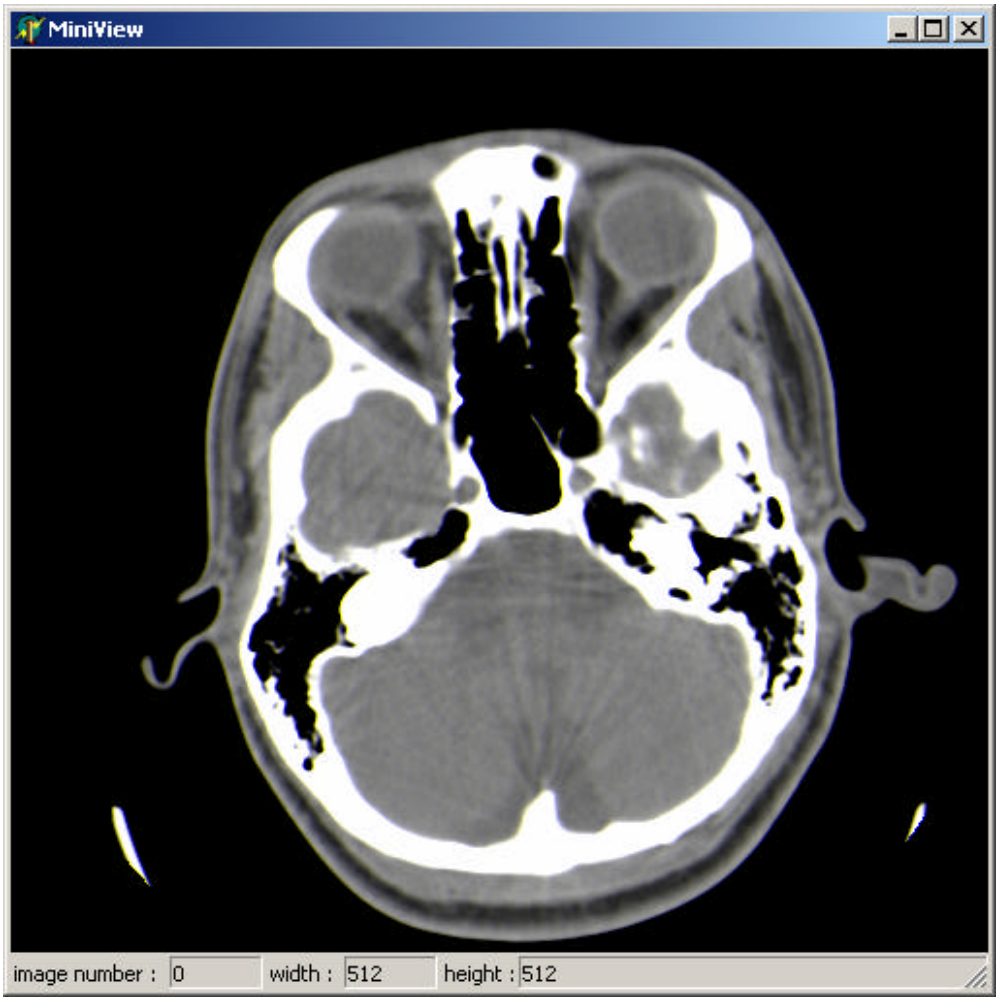


Figura 11.5 Mini Viewer

11.6 Query Tool

Como ditto anteriormente, uma das principais características necessárias para um bom cliente de imagens DICOM, é a possibilidade do mesmo de se comunicar com repositórios de imagens e/ou com os equipamentos de captura de imagens. Na primeira versão do cyclops personal tal funcionalidade ainda não tinhas sido implementada e hoje se encontra funcionando de maneira estável porém sem a capacidade ainda de sincronia com os equipamentos de captura de imagens.

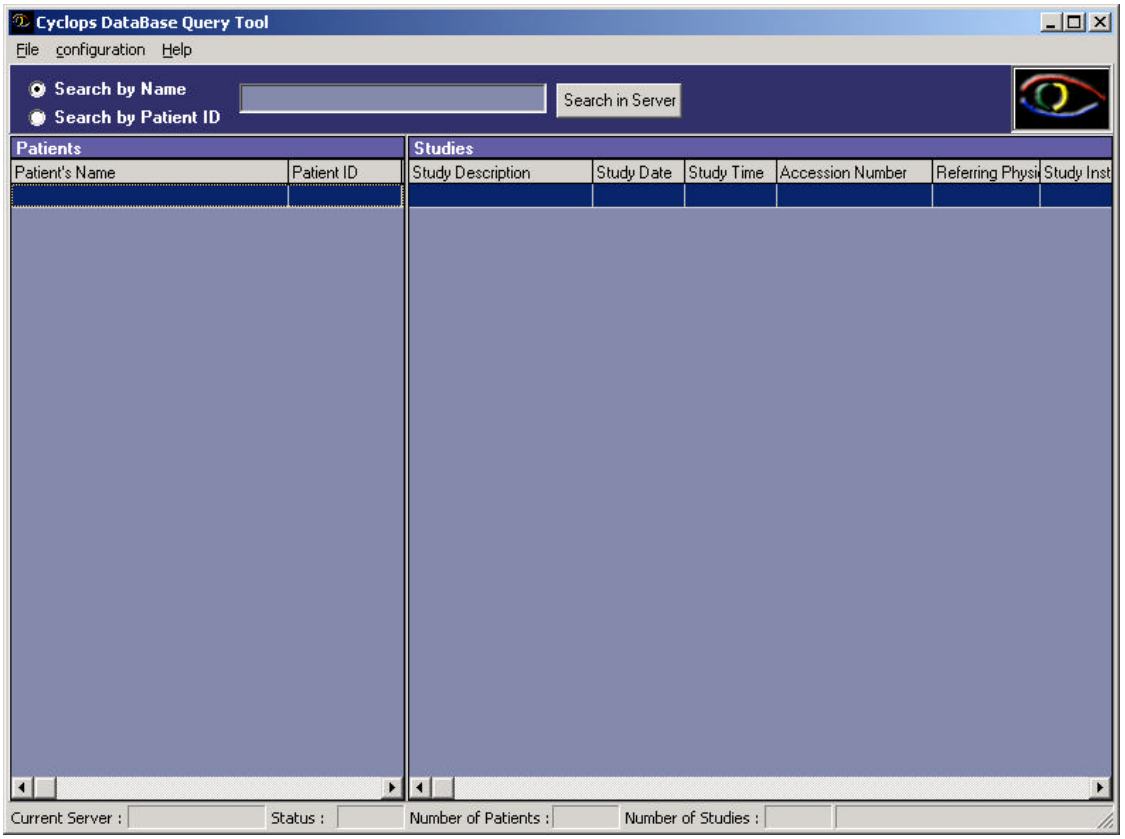


Figura 11.6 Query Tool

12 Implementação e Dados Técnicos

Como dissemos anteriormente, todo o software foi desenvolvido utilizando a linguagem de programação Delphi versão 6.0. Nesta seção apresentaremos algumas das características implementadas no sistema, que cremos, serem de maior pertinência. Infelizmente, devido a extensão do projeto tornasse impossível comentar todos os aspectos da implementação, porém caso algum dado específico seja de interesse do leitor, por favor, reporte-se a documentação contida no CD que acompanha ente trabalho.

12.1 Estrutura da Aplicação

A aplicação foi dividida em vários módulos, ou componentes de forma que características similares ou interdependentes fossem agrupadas, e também para facilitar o processo de entendimento do projeto para outras pessoas que não os desenvolvedores. A Figura 12.1 representa o diagrama de componentes e suas respectivas inter-relações.

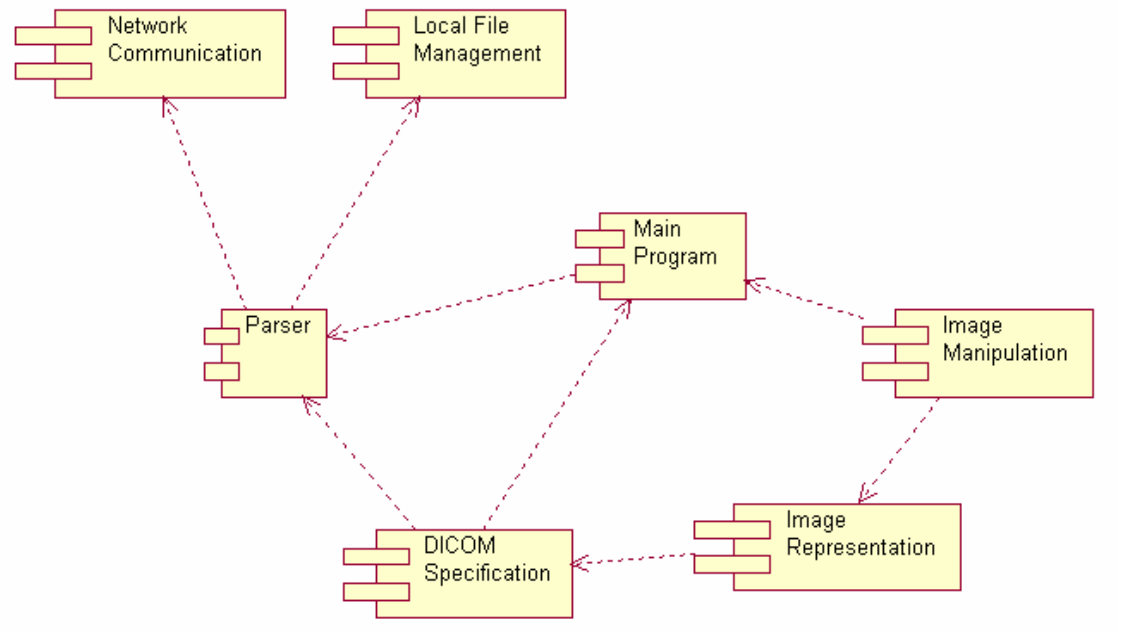


Figura 12.1 Diagrama de componentes

Como podemos observar na Figura 12.1, o sistema esta dividido em 7 componentes. Eles abstraem toda a complexidade do sistema e também possuem suas interdependências representadas. A seguir apresentaremos uma breve explicação de cada um dos componentes:

- Main Program – Neste componente se encontram as classes que representam a parte básica do sistema. A principal delas é a DicomEditorMain que representa a tela principal do sistema. É a partir deste componente que podemos perfazer as operações principais do sistema tal como carregar uma série de imagens da rede ou localmente, lançar qualquer dos módulos para gerenciamento e manipulação de imagens, etc.

- Network Communication – Conjunto de classes responsável pela comunicação via internet. Seu funcionamento básico é buscar imagens de servidores remotos. Este módulo é acessado através da interface Query Tool.

- Local File Management – Conjunto de classes responsável por acessar arquivos .dcm locais ao host onde o aplicativo está rodando, e também pelo gerenciamento dos arquivos temporários das imagens carregadas.

- Parser – Principal e mais complexo componente da aplicação. Responsável por decodificar a informação no formato DICOM e montar a estrutura de objetos DICOM necessários para o funcionamento do programa.

- Image Representation – Componente responsável pela representação da informação visual dos exames em formato reconhecível pelo computador. Perfaz conversões e ajustes.

- Image Manipulation – Todo o conjunto de classes e ferramentas ligadas direta ou indiretamente com a manipulação da informação médica. Fazem parte deste componente o ImageViewer, printer tool, mailer, etc.

- Dicom Specification – A representação da informação médica em uma estrutura de objetos.

12.2 Diagrama de Interfaces do Projeto

O diagrama de interfaces do projeto é uma representação de que interfaces podem ser invocadas a partir de que interfaces. A relação de invocação está sempre representada de baixo para cima, ou seja, se uma classe se encontra no nível 2 e outra no nível 1 significa que a primeira referida pode ser invocada a partir da segunda, mas nunca o contrário.

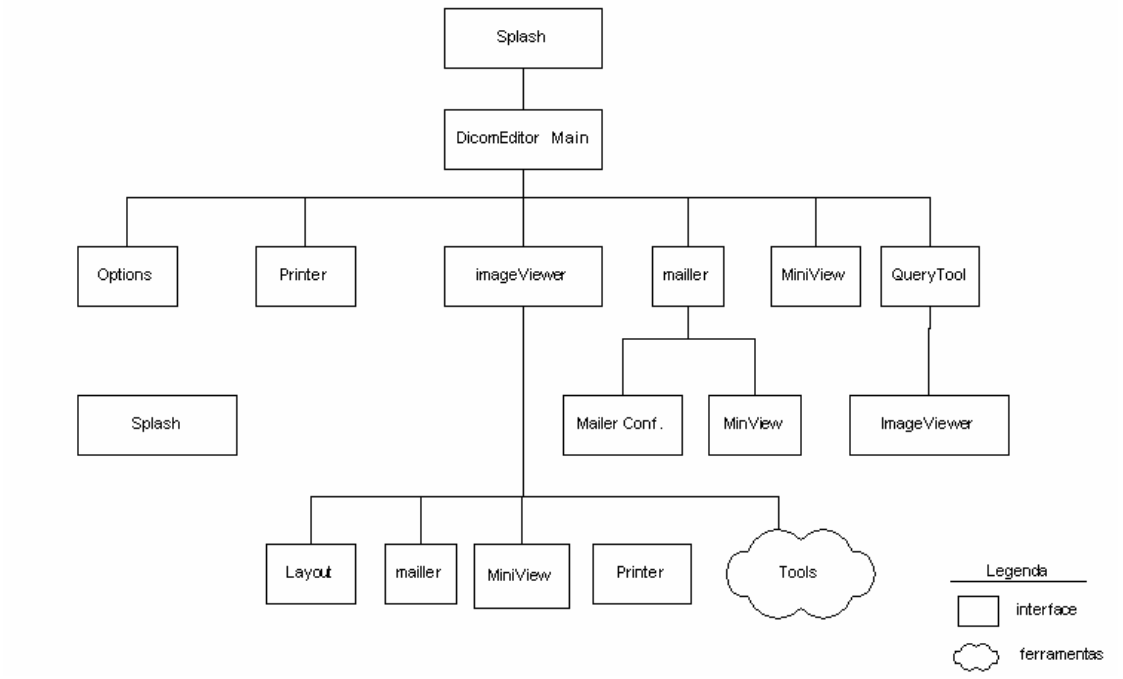


Figura 12.2 Diagrama de interfaces

12.3 Diagrama de Classes (Simplificado)

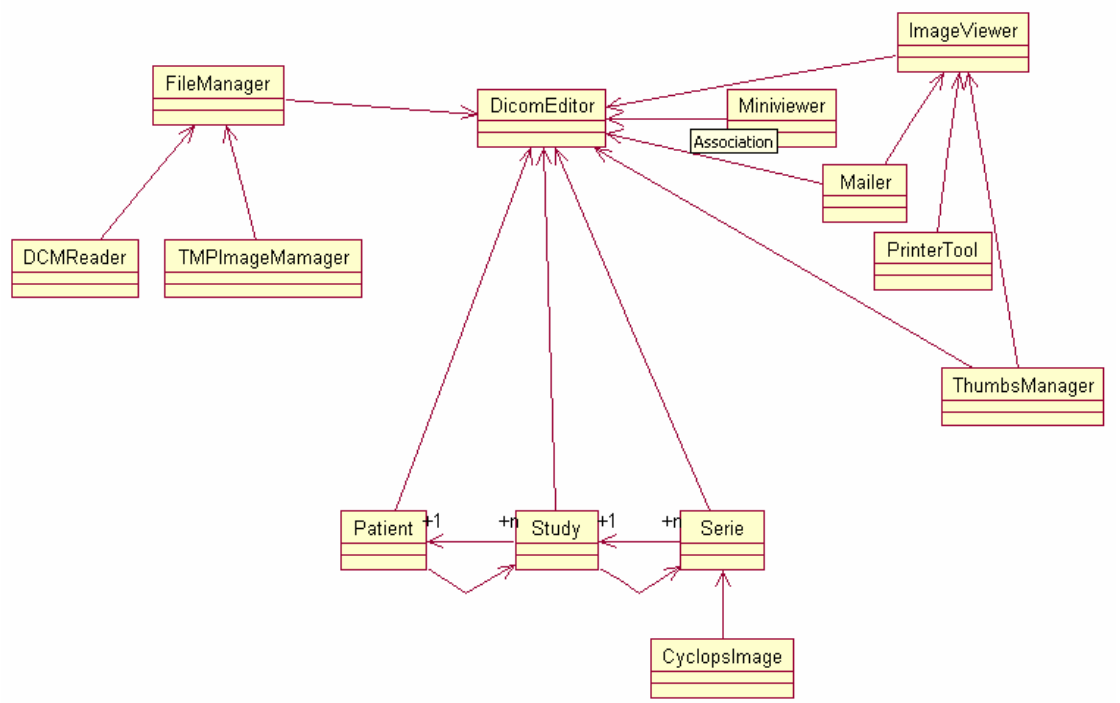


Figura 12.3 Diargama de classes simplificado

12.4 Implementação da conversão HU-RGB

Aqui discutiremos como foi implementada a conversão dos valores de pixel das imagens de HU para RGB. A fundamentação teórica deste item pode ser encontrada em 9.13.

O Exemplo 12-1 apresenta o principal método para a conversão. De imagens representadas e valores HU para RGB. O seguinte método é subdividido em 5 partes:

- Declaração de variáveis
- Teste do tipo de imagem
- Cálculo do mapeamento HU -> RGB
- Passada por todos os pixels da imagens e criação do bmp.
- Retorno da imagem

```
1.  function TCyclopsImage.calcAllImageHUValues;
2.  type
3.  bytearr = array [0..0] of byte;
4.  var
5.  i,
6.  j,
7.  min,
8.  max                 : integer;
9.  v                   : integer;
10. pixValue            : SmallInt;
11. k                       : real;
12. RGBValues           : array of byte;
13. fbmp                : TBitmap;
14. hpal                : HPALETTE;
15. pe                      : PALETTEENTRY;
16. p2                      : array[0..262143] of Byte;

17. begin
18. if Self.Series.generalSeries.modality <> 'US' then //window isn´t
19. begin
20. fbmp:= TBitmap.Create;
21. fbmp.PixelFormat := pf8bit;
22. fbmp.Width  := originalWidth;
23. fbmp.Height := OriginalHeight;

24. //calc min and max HU values
25. min:= WindowCenter - (WindowWidth div 2);
26. max:= WindowCenter + (WindowWidth div 2);
27. k:= (WindowCenter / WindowWidth) - 0.5;
28. setLength(RGBValues, max - min + 1);
29. ////////////////////////////////////////
30. //create a new palette
31. pal := nil;
32. GetMem(pal, sizeof(TLogPalette) + sizeof(TPaletteEntry) * (256));
33. pal.palVersion := $300;
34. pal.palNumEntries := 256;

35. //create a gray scale palette
36. for i := 0 to 255 do
37. begin
38. GetPaletteEntries(fbmp.Palette,i,1,pe);
39. pal.palPalEntry[i].peRed := i;
40. pal.palPalEntry[i].peGreen := i;
41. pal.palPalEntry[i].peBlue := i;
42. end;
43. hpal := CreatePalette(pal^);
44. if hpal <> 0 then
45. fbmp.Palette := hpal;
46. ////////////////////////////////////////
47. //generate a mapped buffer HU  -> RGB
48. for i:= 0 to max - min do
49. RGBValues[i]:= trunc((((min + i)/WindowWidth) - k) * 256);

50. ////////////////////////////////
51. for j:= 0 to OriginalHeight - 1 do
52. for i:= 0 to originalWidth - 1 do
53. begin
54. pixValue:= pixelValue[i,j];
55. if pixValue <= min then v:= 0   else
56. if pixValue >= max then v:= 255 else
57. v:= RGBValues[pixelValue[i,j] - min];
58. //it´s inside the window, let´s get its GrayScale value;
59. p2[j +(i * originalWidth)] := v;

60. end;
61. ////////////////////////////////
62. //Direct memory access using GDI
63. SetBitmapBits(fbmp.Handle,originalWidth*OriginalHeight,@p2);
64. ////////////////////////////////
65. end;
```

```
66.     result := fbmp;
67.     end;
```

Exemplo 12-1 Função para converção HU -> RGB

12.4.1 Declaração de variáveis

Entre as linhas 2 e 16 se encontram a declaração de todas as variáveis do método. A medida que formos explicando o método a função de cada variável ficará mais clara.

12.4.2 Teste do tipo de imagem

Na linha 18 é feito um teste para ver se a imagem não é do tipo US (Ultra-sonografia). Em ultra-sonografias não se aplica conversões de window, que é o que nop final faremos neste método além delas serem coloridas ao contrário do que acontece com as CTs (Tomografias Computadorizadas) e MR (Ressonâncias Magnéticas).

Caso a imagem não seja do tipo US, o método segue.

12.4.3 Cálculo do mapeamento HU -> RGB

Nas linhas 25 a 28 48 e 49 calcula-se o mapeamento dos valores HU para RGB. O que estas linhas fazem é definir o mínimo e máximo valores que serão apresentados para uma dada figura (ou seja, os valores que estão dentro de seu window) e então calcular o valor RGB correspondente a cada valor representável do window.

12.4.4 Criação da paletta

Nas linhas 20,21 e 32 a 45 uma paletta de escalas de cinza é criada. A mesma também é atribuída a imagem que será retornada.

12.4.5 Criação do bmp.

Entre as linhas 51 e 59 são percorridos todos os pixels da imagem aplicando-se a formula de ceifa e substituição apresentada anteriormente, e um buffer vai sendo construído na memória contendo a imagem convertida.

Na linha 63, o buffer contendo a imagem é atribuído ao bitmap e desta forma a imagem está efetivamente convertida.

12.4.6 Retorno da imagem

Na linha 66 a imagem criada é finalmente retornada.

12.4.7 Considerações

O método descrito faz parte do processo de parse da informação DICOM visto que ele nada mais faz do que converter valores HU para RGB.

Este método era um dos principais gargalos do sistema visto que o volume de processamento nele contido era considerável. A utilização da primitiva da GDI na linha 63 melhorou em muito a performance do método. No primeiro protótipo foi utilizado o mecanismo do próprio Delphi para criar a imagem, que consistia em setar pixel a pixel imagem ao invés de copiar um bloco da memória, e como é de se esperar este último forneceu melhores resultados. A diferença de performance é gritante e pode esr observada nos resultados do trabalho.

O modelo antigo de converção pode ser observado no Exemplo 12-2.

```
...
                    for i:= 0 to originalWidth - 1 do
                      for j:= 0 to originalHeight - 1 do
                      begin
                      pixValue:= pixelValue[i,j];
                      if pixValue <= min then v:= 0   else
                      if pixValue >= max then v:= 255 else
                                    v:= RGBValues[pixelValue[i,j] - min];
                                    //it´s inside the window, let´s get its GrayScale value;
                      bmp.Canvas.Pixels[j,i] := v or (v shl 8) or (v shl 16);
                      end;
  end;
...
```

Exemplo 12-2 Maneira antiga de criar a imagem

12.5 Parse dos arquivos DCM

O método apresentado no Exemplo 12-3, é um dos principais métodos do parse dos arquivos .dcm. Basicamente ele recebe o caminho de um arquivo e cria uma instância da classe CyclopsDCMReader, a qual pulará o preâmbulo do arquivo

interpretará a cabeçalho do mesmo e extrairá o tipo de sintaxe de transferência que significa como os bytes devem ser interpretados, ou seja, qual o bit que deve ser considerado o mais significativo, se o mais da esquerda ou o mais da direita.

```
Function TcyclopsFileManager.openDCMFile(filename: string; fastRead: boolean): TCyclopsImage;
var
 group   : TCyclopsDataElementsList;
 patient : TCyclopsPatient;
 study   : TCyclopsStudy;
 series  : TCyclopsSeries;
 image           : TCyclopsImage;
 patientExists: boolean;
begin
 DCMFileReader:= TCyclopsDCMReader.createWithFile(fileName, fastRead);
 DCMFileReader.getDataElements;
 //generate group of patients tags

 group:= DCMFileReader.DataElements.returnSubGroup($0010);
 patient := addPatientWithDEs(group, patientExists);
 patient.Files.Add(fileName);

 //generate group of studies tags

 group:= DCMFileReader.DataElements.returnSubGroups([$0008, $0010, $0020]);
 study := addStudyWithDEsToPatient(group,patient);
 study.Files.Add(fileName);

 //generate group of series tags

 group:= DCMFileReader.DataElements.returnSubGroups([$0020, $0008, $0018]);
 series := addSerieWithDEsToStudy(group,study);
 series.Files.Add(fileName);

 //generate group of images tags

 group:= DCMFileReader.DataElements.returnSubGroups([$0008, $0020, $0028, $7FE0]);
 image:= addImageWithDEsToSerie(group,series, fastRead);
 image.DataElements:= DCMFileReader.DataElements;
 image.FileName:= fileName;
 image.deleteDataElements;
```

```
  //close the file
  DCMFileReader.finalizeInstance;
  DCMFileReader.Free;

  result:= image;
end;
```

Exemplo 12-3 Principal método do Parser

O método getDataElements extrairá todos os tags do arquivo, inclusive o imagem condificada em valores HU.

Depois disso, um tag específico que representa cada classe de informações (pacientes, estudos e séries) é passado para o método returnSubGroup. Ele retornará uma coleção contendo apenas os tags relativos a classe de informação desejada. Esta coleção será passada para o objeto representativo do modelo DICOM que o interpretará e irá adiciona-lo a estrutura de objetos DICOM.

O Exemplo 12-4 mostra a parte do parse de arquivos .dcm que extrai informações relativas as imagens.

```
procedure TCyclopsImage.processDataElement(
  element: TCyclopsDataElement);
begin
  case element.GroupNumber of
    $0008:
      case element.ElementNumber of
        0000:
      end;
    $0020:
      case element.ElementNumber of
        $0013: imageNumber := strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
      end;
    $0028:
      case element.ElementNumber of
        $0002:SamplesPerPixel:= TCyclopsGeneralFunctions.infFromByteArrayLittleEndian(element.Value);
        $0010:originalHeight:= TCyclopsGeneralFunctions.infFromByteArrayLittleEndian(element.Value);
        $0011:originalWidth:= TCyclopsGeneralFunctions.infFromByteArrayLittleEndian(element.Value);
        $1050:begin
            WindowCenter:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            OriginalWindowCenter:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            end;
        $1051:begin
            WindowWidth:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            OriginalWindowWidth:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            end;
      end;
    $7FE0:
      case element.ElementNumber of
        $0010: processImageInfoElement(element);
      end;
  end;
end;
```

Exemplo 12-4 Parte do parse que extrai informações relativas a imagem

12.6 Criação de e-mails usando mime type

12.7 Impressão

```
procedure TCyclopsImagesPrinter.PrintImagesFromHeight(images: TList;
  startHeight: single; var unprintedImages: TList);
var
  done: boolean;
  i,j, k: integer;
  x, y: single;
  imgsWidth, imgsHeight: single;
  ind: integer;
  img: TImage;
begin
  unprintedImages:= TList.Create;

  //this method must only be called when you already have started a printing job
  if not printing then
    exit;

  if images.Count = 0 then //no images to print, nothing to do...
    exit;

  done:= false;

  //get the images dimensions on the paper
  img:= images.first;
  imgsWidth:= (writeableWidth - ((imagesPerLine - 1) * imagesXSpacing)) / imagesPerLine;
  imgsHeight:= (imgsWidth / img.Picture.Bitmap.Width) * img.Picture.Bitmap.height;

  y:= startHeight; //this will be the Y position of the first row of Images

  //i means the printing row (starting at 0);
  for i:= 0 to images.Count div imagesPerLine do
    begin
    x:= leftMargin; //this will be the X position of the first column of Images
    for j:= 0 to imagesPerLine - 1 do
      begin
      ind:= (i * imagesPerLine) + j; //this is the index of the image to be printed now

      if (ind > (images.Count - 1)) then
      //if index is greater, all images have been printed and the job is done...
        begin
        done:= true;
        break;
        end // end if
      else
      //... if not, get the TImage and print it!
```

```
                    img:= images.items[ind];

                    printer.PrintGraphic(x, y, imgsWidth, imgsHeight, img);
                    //prints the image "img" at (X,Y) , with the calculated dimensions;

                    x:= x + imgsWidth + imagesXSpacing;
                   //pushes X to the right of the last printed image,
                    //giving the right spacing between them

                    end; //end for j:=... (row printing)

                  if done then //no more images to be printed, finished!
                    break
                  else
                    begin
                    y:= y + imgsHeight + imagesYSpacing;
                    //pushes Y to the bottom of the last printed row,
                    //giving the right spacing between them

                    if ((y + imgsHeight) > (paperHeight - BottomMargin)) then
                    //this means that the new row would be beyond the bottom margin
                    //if so, put all the unprinted images in unprintedImages, and exit method
                      begin
                      for k:= ind  + 1 to images.Count- 1 do
                        unprintedImages.Add(images.Items[k]);
                      break;
                      end; //end if (bottom margin check)
                    end;// end else (if done)
                  end; //enf for i:= ... (new rows);
end;
```

Exemplo 12-5 Fragmento de código que gera o relastório de impressão

12.8 Piloto

```
procedure TPiloto.montaImagemHor;
var
    x,y                                 : integer;
  im                        : tImage;
  cont                      : integer;
  offset                    : integer;
  offsetAcc     : integer;
  P                         : PByteArray;
begin
    //inicializações do método
  offsetAcc := 1;

  //laço para a coleta das linhas de todas as imagens da série
  for cont := 0 to colImage.Count - 1 do
  begin
      im := colImage.Items[cont];           //imagem de referencia
    offsetAcc := offsetAcc - 1;
    P := im.Picture.Bitmap.ScanLine[pos_ref];
    for offset := 0 to thickness - 1 do
    begin
      for x := 0 to 511{im.Width} do
      begin
        Picture.Bitmap.Canvas.Pixels[x,cont + offsetAcc] := P[x];

      end;
      inc(offsetAcc);

      end;
    end;
end;
```

Exemplo 12-6geração de imagens piloto

Figura 12.4 imagem piloto gerada

12.9 Window

```
procedure TCyclopsImage.changePalette(newWindowCenter,NewWindowWidth: integer);
type
            bytearr = array [0..0] of byte;
var
  i, j, min, max: integer;
  v                                         : integer;
  pixValue              : SmallInt;
  k                                         : real;
  RGBValues                       : array of byte;
  p                                         : ^bytearr;
  p2                                : array[0..262143] of Byte;

begin

   if Self.Series.generalSeries.modality <> 'US' then //window isn´t
   begin
      WindowCenter:= newWindowCenter; //adjust window center and
      WindowWidth:= newWindowWidth;   //                                                width


     min:= WindowCenter  - (WindowWidth div 2);
     max:= WindowCenter + (WindowWidth div 2);
     k:= (WindowCenter / WindowWidth) - 0.5;
     setLength(RGBValues, max - min + 1);

     for i:= 0 to max - min do
       RGBValues[i]:= trunc((((min + i)/WindowWidth) - k) * 256);

     ////////////////////////////
     for j:= 0 to OriginalHeight - 1 do
       for i:= 0 to originalWidth - 1 do
       begin
          pixValue:= pixelValue[i,j];
          if pixValue <= min then v:= 0   else
          if pixValue >= max then v:= 255 else
```

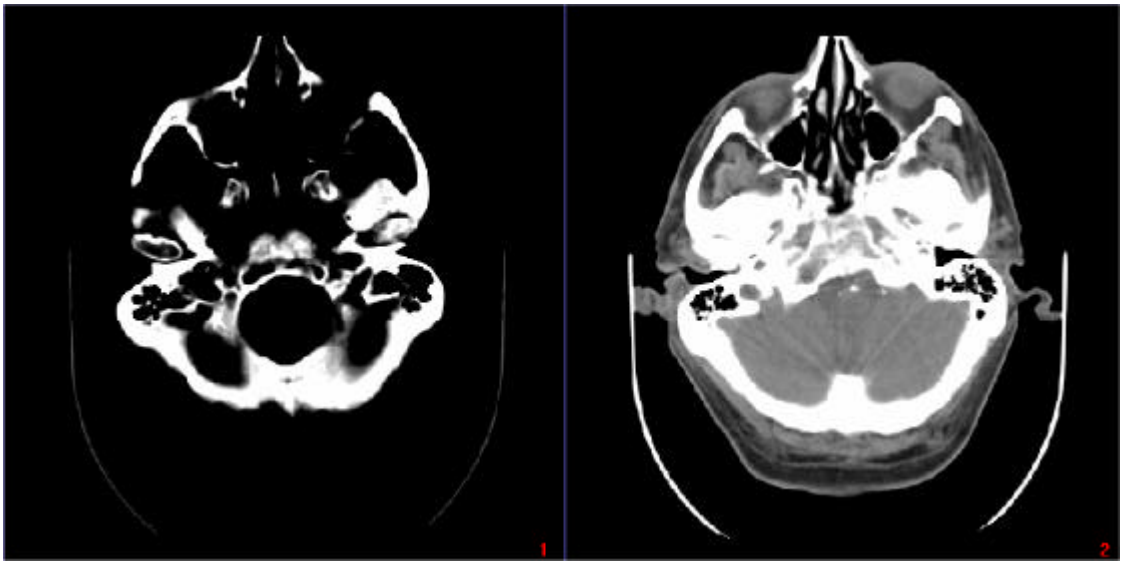
```
                        v:= RGBValues[pixelValue[i,j] - min];
                        //it´s inside the window, let´s get its GrayScale value;
          p2[j +(i * originalWidth)] := v;

       end;
     ////////////////////////////
     SetBitmapBits(Picture.Bitmap.Handle,originalWidth*OriginalHeight,@p2);

              FreeMemory(@p);
     Refresh;
     ////////////////////////////
   end;


end;
```

Exemplo 12-7 Fragmento de código que gera a nova paletta



Exemplo 12-8 Imagens com window alterado e normal, respectivamente

13 Considerações sobre Performance

Devido ao fato deste projeto tratar de informações que são apresentadas em grandes volumes algumas considerações a respeito dos requisitos de software e hardware devem ser feitas visando o bom funcionamento do mesmo. Muitos testes foram executados, como podemos ver no item 14 deste trabalho. Nestes testes foi testado o funcionamento do projeto em várias versões do windows e também em várias configurações de hardware. Variou-se a quantidade de memória RAM disponível nos programas e, desta forma, chegou-se a uma definição mínima e recomendada para o bom funcionamento do projeto.

13.1 Requisitos de S/H

Quanto aos requisitos de software o programa requer o sistema operacional windows funcionando em todas as suas versões a partir do Windows 95. A versão do projeto para Linux ainda se encontra muito instável e com alguns problemas na manipulação de imagens visto que esta é a parte não portável do projeto sendo fortemente dependente ao sistema operacional.

Quanto ao Hardware necessário, é desejado um computador modelo IBM-PC com monitor colorido e mouse. A velocidade do barramento da placa mãe influencia enormemente no desempenho do sistema visto que um dos grandes gargalos do sistema é o processo de abertura/leitura dos arquivos .dcm. Outro fator que também influencia a performance do projeto é a velocidade de acesso da memória e ao disco rígido.

13.2 Configuração Mínima

Após o período de testes, definiu-se como configuração mínima:

- Processador Celeron 333MHz
- 64 MB de memória RAM
- 300 MB em espaço no disco rígido
- monitor colorido SupoerVGA 14''
- mouse
- teclado

A velocidade do processador de 333MHz foi a mínima aceitável visto que alguns processos como a geração de imagens piloto e conversão de imagens HU -> RGB são processos que requisitam muito poder de processamento.

A quantidade de memória mínima de 64MB foi verificada visto que o sistema operacional absorve boa parte da mesma e projeto requisita de grande quantidade de memória para seu bom funcionamento. Mesmo assim para uma série simples de por exemplo uma tomografia de 20 imagens, fatalmente fará swap em disco rígido.

13.3 Configuração Recomendada

Após o período de testes, definiu-se como configuração recomendada:

- Processador Athon 1.1GHz ou PentiunIII 1GHz
- 256 MB de memória RAM
- 700 MB em espaço no disco rígido
- monitor colorido SupoerVGA 17''
- mouse
- teclado

Com um processador mais potente o software requisitará menos recursos do computador além de permitir que outros programas sejam executados concorrentemente. Com uma quantidade maior de memória, a possibilidade de swap será menor melhorando assim a performance do sistema.

14 Resultados

Ao final do mês de março uma Segunda versão estável do software foi lançada. Como já dissemos no histórico do projeto, esta versão contém todas as funções existentes na versão anterior que foram validadas, expandidas e melhoradas além de contar com o mecanismo de comunicação com bases de dados de imagens.

A ferramenta de e-mail foi finalmente terminada e a geração de imagens piloto foi criada efetivamente.

Muitas das classes de suporte necessárias para a migração das outras aplicações do projeto Cyclops foram criadas e o projeto agora se encontra em sua terceira fase de desenvolvimento, onde o problema de sincronia do cyclops personal com os equipamentos de captura de imagens será resolvido.

14.1 Plataforma de Testes

Para o período de testes foi utilizado o centro de estudos do hospital universitário, onde são ministradas aulas de radiologia e diagnóstico radiológico auxiliado por computador.

O projeto também foi testado na clínica DMI (Diagnóstico Médico por Imagens), onde com o auxílio do Dr. Felipe Nobre obtivemos valorosas informações quanto a performance do projeto, grau de usabilidade e informações médicas.

14.2 Validação do Software

A parte técnica do software está sendo validada constantemente pelo corpo de desenvolvedores do projeto cyclops do Brasil, que periodicamente testa se as novas funcionalidades estão de acordo com a especificação DICOM e similares as existentes no projeto Cyclops(versão Smalltalk). Os médicos da clinica DMI, em especial o Dr. Felipe

Nobre estão sendo de grande auxílio perante a validação do software sob uma ótica médica.

14.3 Utilização do Software

De posse da segunda versão estável do projeto, o mesmo começa agora a ser utilizado na clinica DMI e por alguns professores do Hospital Universitário em substituição ao DICOM Editor original e ao software e-film.

Para o mês de maio, espera-se instalar o software no laboratório de telemedicina do hospital universitário a fim de que seja utilizado durante algumas aulas de radiologia, onde hoje o DICOM Editor é utilizado.

14.4 Comparações

Durante o processo de validação do sistema vários testes de performance foram executados, dentre eles o que podemos destacar como mais interessante foram os relativos a performance em tempo de abertura das imagens. A seguir poderemos ver duas baterias de teste, executadas em dois computadores distintos e também com diferentes volumes de memória RAM. No primeiro teste testou-se o sistema em um Pentium Celeron 333MHz com 256,128 e 64 MB de RAM respectivamente, utilizando-se a primeira versão do projeto, versão esta que não manipulava as imagens utilizando a GDI do windows. Em seguida testamos a mesma versão do projeto (versão um sem o uso da GDI) em um Athon 1.3 MHz com 256,128,64 de memória RAM.

A segunda bateria de testes também utilizou os mesmos computadores porém com a nova versão do projeto.

14.4.1 Primeira bateria de testes

Esta bateria de teste foi executada utilizando a primeira versão do projeto.

no imagens	tempo (seg)		
5	1	1	3
10	2	2	6
15	3	4	9
20	4	6	12
25	5	8	16
30	7	11	20
35	9	14	24
40	11	17	29
45	13	21	34
50	16	25	40

Tabela 14-1Execução do sistema em um computador Celeron 333MHz com 256,128 e 64 MB

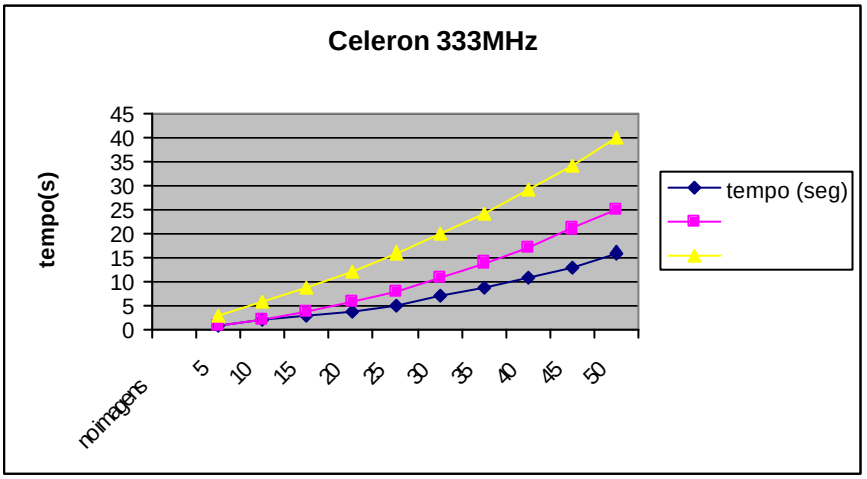


Figura 14.1 Resultados do teste com o projeto v. 1

no imagens	tempo (seg)		
5	1	1	1
10	1	1	2
15	1	2	3
20	2	3	4
25	2	4	5
30	3	5	7
35	4	6	9
40	5	8	11
45	6	10	13
50	7	12	15

Tabela 14-2 1Execução do sistema em um computador Athon 1.3GHz 256,128 e 64 MB

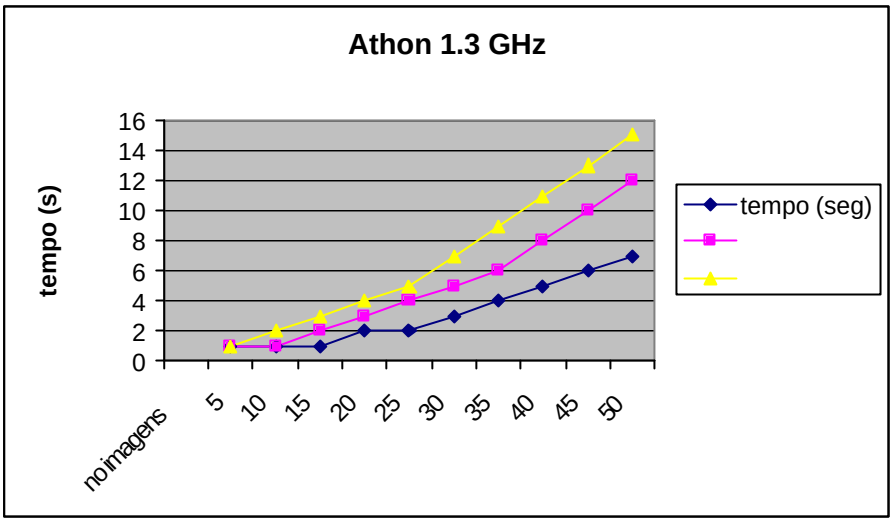


Figura 14.2 Resultados do Teste com o projeto v. 1

14.4.2 Segunda bateria de testes

Esta bateria de teste foi executada a segunda versão do projeto.

no imagens	tempo (seg)		
5	1	1	2
10	1	2	5
15	3	3	8
20	4	5	11
25	5	7	15
30	7	9	19
35	9	11	23
40	11	14	28
45	13	18	33
50	15	22	39

Tabela 14-3 Execução do sistema em um computador Celeron 333MHz 256,128 e 64 MB

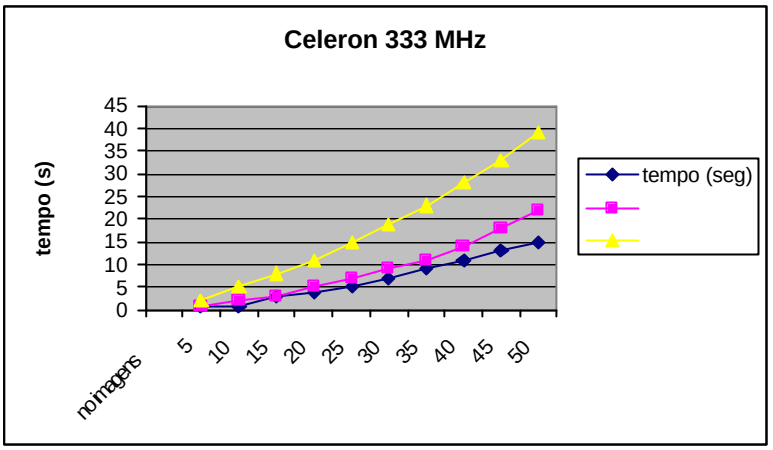


Figura 14.3 Resultados do Teste com o projeto v. 2

no imagens	tempo (seg)		
5	1	1	1
10	1	1	2
15	1	2	3
20	2	2	4
25	2	3	6
30	3	4	8
35	3	4	11
40	4	5	15
45	4	7	20
50	5	9	26

Tabela 14-4 31Execução do sistema em um Atlhon 1.3GHz 256,128 e 64 MB

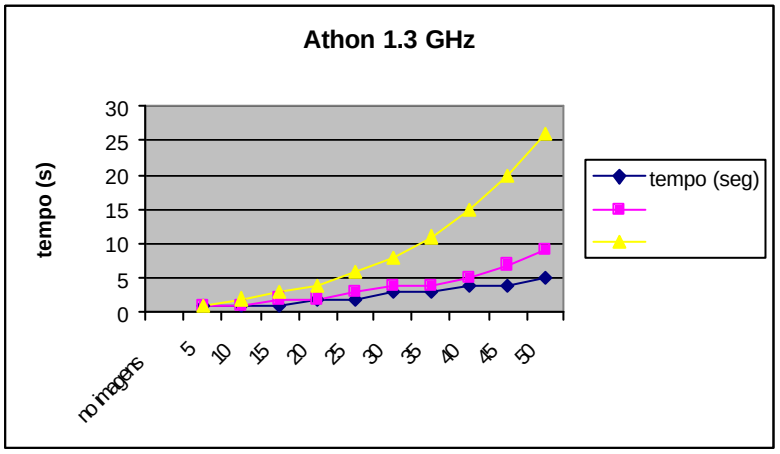


Figura 14.4 Resultados do Teste com o projeto v. 2

Como podemos observar nos gráficos acima apresentados, o tempo para o carregamento de imagens cresce em geral exponencialmente em relação ao número de imagens a serem abertas.

15 Trabalhos Futuros

Ao termino do segundo esforço de desenvolvimento deste projeto, esforço este que representa a elaboração deste trabalho, podemos verificar que bons resultados foram alcançados. Mas para fazer do Cyclops Personal uma ferramenta de qualidade internacional muito ainda pode-se fazer. Dentre as pendências imediatas do projeto podemos citar:

- Dicom Stack Communication
- Cyclops Cyne
- Multiframe Support
- Concordância com o padrão

16 Conclusão

Ao termino do prazo para o lançamento da segunda versão do cyclops personal, muitas das pendências existentes na primeira versão do mesmo foram sanadas e várias outras funcionalidades foram acrescentadas, cumprindo desta forma os objetivos gerais e específicos deste projeto.

A gama de objetos de suporte a compatibilidade que foram criados durante o período de desenvolvimento desta segunda versão do software foi validada e está em concordância ao esperado, e já se inicio o processo de migração de outras aplicações do projeto Cyclops para o DELPHI, dentre elas podemos citar o Ajuste do Atlas de Talairah e o calculo do diâmetro de stents utilizados na operação de correção do aneurisma da artéria aorta.

Assim, é mais um passo que o Projeto Cyclops dá para a implementação prática de suas muitas soluções. Porém, vale lembrar que ainda é necessário muito trabalho para que este software possa ser seguramente utilizado no meio hospitalar.

17 Apêndice A – Lista de classes

1. UcyclopsContrastBolusModule – Pertencente ao modelo de objetos do DICOM
2. UCyclopsCTImageModule - Pertencente ao modelo de objetos do DICOM
3. uCyclopsDataElement - Responsável pelo parse dos arquivos dcm
4. uCyclopsDataElementsList - Responsável pelo parse dos arquivos dcm
5. UCyclopsDBObject – Pertencente ao modelo de objetos do DICOM
6. uCyclopsDCMReader –
7. uCyclopsDICOMBETransferSyntax –
8. UCyclopsDicomCTImage – Pertencente ao modelo de objetos do DICOM
9. UcyclopsDicomImage – Pertencente ao modelo de objetos do DICOM
10. uCyclopsDICOMImplicitLETransferSyntax –
11. uCyclopsDICOMLETransferSyntax –
12. UCyclopsDicomMRImage – Pertencente ao modelo de objetos do DICOM
13. UCyclopsDicomNMImage – Pertencente ao modelo de objetos do DICOM
14. UCyclopsDicomSCImage – Pertencente ao modelo de objetos do DICOM
15. uCyclopsDICOMTransferSyntax –
16. UCyclopsDicomUSImage – Pertencente ao modelo de objetos do DICOM
17. UCyclopsDicomXRAYImage – Pertencente ao modelo de objetos do DICOM
18. UcyclopsEquipment – Pertencente ao modelo de objetos do DICOM
19. uCyclopsFileManager –
20. UcyclopsFindings –
21. UcyclopsFrameOfReference – Pertencente ao modelo de objetos do DICOM

22. uCyclopsGeneralFunctions –
23. uCyclopsGeneralSeriesModule – Pertencente ao modelo de objetos do DICOM
24. uCyclopsGeneralStudyModule – Pertencente ao modelo de objetos do DICOM
25. UCyclopsIEDefinitions – Pertencente ao modelo de objetos do DICOM
26. uCyclopsPatient – Pertencente ao modelo de objetos do DICOM
27. UcyclopsPatientModule – Pertencente ao modelo de objetos do DICOM
28. UcyclopsPatientStudyModule – Pertencente ao modelo de objetos do DICOM
29. uCyclopsSeries – Pertencente ao modelo de objetos do DICOM
30. uCyclopsStudy – Pertencente ao modelo de objetos do DICOM
31. UcyclopsImageList –
32. UcyclopsTagsDictionary –
33. uCyclopsTempImagesManager –
34. ImageLibrary –
35. LineLibrary –
36. UcyclopsImage – Pertencente ao modelo de objetos do DICOM
37. UpaintBoxEx –
38. VectorGraphicsListLibrary –
39. VectorGraphicsNodeLibrary –
40. fSplash –
41. PrintDr1 –
42. PrintDr2 –
43. UcyclopsCyne –
44. uCyclopsLoadProgressForm –

45. UDICOMEditorMain –
46. ufAbout –
47. ufDirectoryFilter –
48. ufOptions –
49. UfprintImages –
50. ufQueryTool –
51. UfrmLayout –
52. UfrmZoom –
53. ufServerStarter –
54. uImageViewer –
55. uMailler
56. UminiView -
57. PrintDrv –
58. uCyclopsImagesPrinter –
59. uCyclopsPrinter –
60. uPaperSheet –
61. CyclopsThumbnail –
62. uThumbnails –
63. uCyclopsDICOMServer –
64. UmaillerOptions –
65. uCyclopsHandle –

18 Apêndice B – Código Fonte

```
//----------------------------------------------------------------------------------------------------
program CyclopsPerssonal;

uses
  Forms,
  UCyclopsContrastBolusMo dule in 'source\DICOM classes\UCyclopsContrastBolusModule.pas',
  UCyclopsCTImageModule in 'source\DICOM classes\UCyclopsCTImageModule.pas',
  uCyclopsDataElement in 'source\DICOM classes\uCyclopsDataElement.pas',
  uCyclopsDataElementsList in 'source\DICOM classes\uCyclopsDataElementsList.pas',
  UCyclopsDBObject in 'source\DICOM classes\UCyclopsDBObject.pas',
  uCyclopsDCMReader in 'source\DICOM classes\uCyclopsDCMReader.pas',
  uCyclopsDICOMBETransferSyntax in 'source\DICOM classes\uCyclopsDICOMBETransferSyntax.pas',
  UCyclopsDicomCTImage in 'source\DICOM classes\UCyclopsDicomCTImage.pas',
  UCyclopsDicomImage in 'source\DICOM classes\UCyclopsDicomImage.pas',
  uCyclopsDICOMImplicitLETransferSyntax in 'source\DICOM classes\uCyclopsDICOMImplicitLETransferSyntax.pas',
  uCyclopsDICOMLETransferSyntax in 'source\DICOM classes\uCyclopsDICOMLETransferSyntax.pas',
  UCyclopsDicomMRImage in 'source\DICOM classes\UCyclopsDicomMRImage.pas',
  UCyclopsDicomNMImage in 'source\DICOM classes\UCyclopsDicomNMImage.pas',
  UCyclopsDicomSCImage in 'source\DICOM classes\UCyclopsDicomSCImage.pas',
  uCyclopsDICOMTransferSyntax in 'source\DICOM classes\uCyclopsDICOMTransferSyntax.pas',
  UCyclopsDicomUSImage in 'source\DICOM classes\UCyclopsDicomUSImage.pas',
  UCyclopsDicomXRAYImage in 'source\DICOM classes\UCyclopsDicomXRAYImage.pas',
  UCyclopsEquipment in 'source\DICOM classes\UCyclopsEquipment.pas',
  uCyclopsFileManager in 'source\DICOM classes\uCyclopsFileManager.pas',
  UCyclopsFindings in 'source\DICOM classes\UCyclopsFindings.pas',
  UCyclopsFrameOfReference in 'source\DICOM classes\UCyclopsFrameOfReference.pas',
  uCyclopsGeneralFunctions in 'source\DICOM classes\uCyclopsGeneralFunctions.pas',
  uCyclopsGeneralSeriesModule in 'source\DICOM classes\uCyclopsGeneralSeriesModule.pas',
  uCyclopsGeneralStudyModule in 'source\DICOM classes\UCyclopsGeneralStudyModule.pas',
  UCyclopsIEDefinitions in 'source\DICOM classes\UCyclopsIEDefinitions.pas',
  uCyclopsPatient in 'source\DICOM classes\uCyclopsPatient.pas',
  UCyclopsPat ientModule in 'source\DICOM classes\UCyclopsPatientModule.pas',
  UCyclopsPatientStudyModule in 'source\DICOM classes\UCyclopsPatientStudyModule.pas',
  uCyclopsSeries in 'source\DICOM classes\uCyclopsSeries.pas',
  uCyclopsStudy in 'source\DICOM classes\uCyclopsStudy.pas',
  UCyclopsImageList in 'source\general classes\UCyclopsImageList.pas',
  UCyclopsTagsDictionary in 'source\general classes\UCyclopsTagsDictionary.pas',
  uCyclopsTempImagesManager in 'source\general classes\uCyclopsTempImagesManager.pas',
  ImageLibrary in 'source\graphic classes\imageLibrary.pas',
  LineLibrary in 'source\graphic classes\LineLibrary.pas',
  UCyclopsImage in 'source\graphic classes\UCyclopsImage.pas',
  UPaintBoxEx in 'source\graphic classes\UPaintBoxEx.pas',
  VectorGraphicsListLibrary in 'source\graphic classes\VectorGraphicsListLibrary.PAS',
  VectorGraphicsNodeLibrary in 'source\graphic classes\VectorGraphicsNodeLibrary.PAS',
  fSplash in 'source\interface classes\fSplash.pas' {fSplashScreen},
  PrintDr1 in 'source\interface classes\PrintDr1.pas' {PreviewForm},
  PrintDr2 in 'source\interface classes\PrintDr2.pas' {PrintBarForm},
  UcyclopsCyne in 'source\interface classes\UcyclopsCyne.pas' {frmCine},
  uCyclopsLoadProgressForm in 'source\interface classes\uCyclopsLoadP rogressForm.pas' {CyclopsLoadProgressForm},
  UDICOMEditorMain in 'source\interface classes\UDICOMEditorMain.pas' {fCyclopsDICOMEditorMain},
  ufAbout in 'source\interface classes\ufAbout.pas' {fAbout},
  ufDirectoryFilter in 'source\interface classes\ufDirectoryFilter.pas' {fDirectoryFilter},
  ufOptions in 'source\interface classes\ufOptions.pas' {fOptions},
  UfprintImages in 'source\interface classes\UfprintImages.pas' {fImagePrint},
  ufQueryTool in 'source\interface classes\ufQueryTool.pas' {fQueryToo l},
  UfrmLayout in 'source\interface classes\UfrmLayout.pas' {frmLayout},
  UFrmZoom in 'source\interface classes\UFrmZoom.pas' {frmZoom},
  ufServerStarter in 'source\interface classes\ufServerStarter.pas' {fServerStarter},
  uImageViewer in 'source\interface classes\uImageViewer.pas' {fImageViewer},
  uMailler in 'source\interface classes\uMailler.pas' {frmMailler},
  UMiniView in 'source\interface classes\UMiniView.pas' {MiniView},
  sak_util in 'source\mailler classes\sak_util.pas',
  SakMIME in 'source\mailler classes\SakMIME.pas',
  SakMsg in 'source\mailler classes\SakMsg.pas',
  SakRegister in 'source\mailler classes\SakRegister.pas',
```

```
  SakSMTP in 'source\mailler classes\SakSMTP.pas',
  PrintDrv in 'source\printer classes\PrintDrv.pas',
  uCyclopsImagesPrinter in 'source\printer classes\uCyclopsImagesPrinter.pas',
  uCyclopsPrinter in 'source\printer classes\uCyclopsPrinter.pas',
  uPaperSheet in 'source\printer classes\uPaperSheet.pas',
  CyclopsThumbnail in 'source\thumbs classes\CyclopsThumbnail.pas',
  uThumbnails in 'source\thumbs classes\uThumbnails.pas',
  uCyclopsDICOMServer in 'source \general classes\uCyclopsDICOMServer.pas',
  SakPOP3 in 'source\mailler classes\SakPOP3.pas',
  UMaillerOptions in 'source\interface classes\UMaillerOptions.pas' {frmMaillerConfig},
  uCyclopsHandle in 'source\general classes\uCyclopsHandle.pas',
  DIMime in 'source\mailler classes\DIMime.pas',
  DIMimeStreams in 'source\mailler classes\DIMimeStreams.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TfSplashScreen, fSplashScreen);
  Application.CreateForm(TfCyclopsDICOMEditorMain, fCyclopsDICOMEditorMain);
  Application.CreateForm(TfrmLayout, frmLayout);
  Application.CreateForm(TfrmMaillerConfig, frmMaillerConfig);
  Application.CreateForm(TfrmMailler, frmMailler);
  Application.Run;
end.

//----------------------------------------------------------------------------------------------------

CLASS   : TfImageViewer
Author    :         Charles I. Wust
Date      :         09/10/2001

Description         :
Atributes :
Methods :
Related Errors      :


Need to by Done     :
}
unit CyclopsThumbnail;

interface

uses
  Windows, Messages, SysUtils, Classes, Controls, ExtCtrls, UCyclopsImage, Graphics;

type
  TCyclopsThumbnail = class(TPanel)
  private
    { Private declarations }
    fSelected: boolean;
    fPicture: TBitmap;
  protected
    { Protected declarations }
    procedure paint; override;
  public
    { Public declarations }
    sourceCyclopsImages: TCyclopsImage;
  published
    { Published declarations }
    property Selected: boolean read fSelected write fSelected default false;
    property Picture:TBitmap read fPicture write fPicture default nil;
  end;

procedure Register;

implementation

procedure Register;
```

```
begin
  RegisterComponents('Cyclops', [TCyclopsThumbnail]);
end;

{ TCyclopsThumbnail }

procedure TCyclopsThumbnail.paint;
var
  tempW: integer;
  tempC: TColor;
begin

  if fSelected then
    begin
    with canvas do
      begin
      tempW:= Pen.Width;
      tempC:= Pen.Color;
      Pen.Color:= clRed;
      Pen.Width:= 10;
      brush.color := clNone;
      brush.Style := bsClear;
      Pen.Mode:= pmCopy;
      Rectangle(Rect(0,0, Width, height));
      Pen.Width:= tempW;
      Pen.Color:= tempC;
      end;
    end;
  inherited;
end;

end.


//----------------------------------------------------------------------------------------------------

unit fSplash;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, uDICOMEditorMain, uCyclopsGeneralFunctions;

type
  TfSplashScreen = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Timer1: TTimer;
    procedure FormActivate(Sender: TObject);
    procedure Timer1Timer(Sender: T Object);
  private

  public
  end;

var
  fSplashScreen: TfSplashScreen;

implementation

{$R *.dfm}

procedure TfSplashScreen.FormActivate(Sender: TObject);
begin
  refresh;
  timer1.enabled:= true;
```

```
end;

procedure TfSplashScreen.Timer1Timer(Sender: TObject);
begin
  timer1.Enabled:= false;
  fCyclopsDICOMEditorMain.Show;
  Hide;
end;

end.


//----------------------------------------------------------------------------------------------------

unit UCyclopsContrastBolusModule;

interface
        type TCy clopsContrastBolusModule = class

private
        _contrastBolusAgent             : string;
    _contrastBolusRoute                 : string;
    _contrastBolusVolume        : string;
    _contrastBolusStartTime     : string;
    _contrastBolusStopTime      : string;
    _contrastBolusTotalDose     : string;
protected

public

published
        property contrastBolusAgent            : string read _contrastBolusAgent write _contrastBolusAgent;
    property contrastBolusRoute         : string read _contrastBolusRoute write _contrastBolusRoute;
    property contrastBolusVolume        : string read _contrastBolusVolume write _contrastBolusVolume;
    property contrastBolusStartTime     : string read _contrastBolusStartTime write _contrastBolusStartTime;
    property contrastBolusStopTime      : string read _contrastBolusStopTime write _contrastBolusStopTime;
    property contrastBolusTotalDose     : string read _contrastBolusTotalDose write _contrastBolusTotalDose;
end;
implementation

end.

//----------------------------------------------------------------------------------------------------

unit UCyclopsCTImageModule;

interface
        type TCyclopsCTImageModule = class

private
    _rescaleIntercept   : string;
    _rescaleSlope               : string;
    _kvp                                : string;
protected

public

published
        property rescaleIntercept       : string read _rescaleIntercept write _rescaleIntercept;
    property rescaleSlope       : string read _rescaleSlope             write _rescaleSlope;
    property kvp                                : string read _kvp                                      write _kvp;
end;
implementation

end.

//----------------------------------------------------------------------------------------------------

unit UcyclopsCyne;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, uCyclopsImageList;

type
  TfrmCine = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Timer1: TTimer;
    ComboBox1: TComboBox;
    Label1: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    BitBtn5: TBitBtn;
  private

  public
                        imageList               : TCyclopsImageList;
  end;

var
  frmCine: TfrmCine;

implementation

{$R *.dfm}

end.

//----------------------------------------------------------------------------------------------------------

unit uCyclopsDataElement;

interface

uses sysUtils, uCyclopsGeneralFunctions;

type TCyclopsDataElement = class
  protected
  fGroupNumber, fElementNumber: word; //responsible for tag
  fVRC1, fVRC2: char;               //responsible for VR
  fValueLength: longword;             //DE Length
  public
  Value: TCyclopsByteArray;
  constructor createWith(vGroupNumber, vElementNumber: word; vVRC1, vVRC2: char;
                vValueLength: longword; var vValue: TCyclopsByteArray);
  function asString: string;
  function tagString: string;

  property GroupNumber: word read fGroupNumber write fGroupNumber;
  property ElementNumber: word read fElementNumber write fElementNumber;
  property VRC1: char read fVRC1 write fVRC1;
  property VRC2: char read fVRC2 write fVRC2;
  property ValueLength:longword read fValueLength write fValueLength;
end;

implementation

{ TCyclopsDataElement }

function TCyclopsDataElement.asString: string;
var
  VR, ret: string;
  i: integer;
begin
```

```
  if (VRC1 = #0) or (VRC2 = #0) then
    begin
     ret:= '(' + inttostr(GroupNumber) + ',' + inttostr(ElementNumber) +
              ') Length = ' + inttostr(valueLength) + ' Value = ';
    end
  else
    begin
     VR:= '' + VRC1 + VRC2;
     ret:= '(' + inttostr(GroupNumber) + ',' + inttostr(ElementNumber) +
              ') VR = ' + VR + ' Length = ' + inttostr(valueLength)  + ' Value = ';
    end;
  for i:= 0 to high(Value) do
    begin
    if (value[i] >= 33) and (value[i] <= 126) then
      //most common chars are in this interval
      ret:= ret + char(value[i])
    else
      ret:= ret + ' ';
    end;
  result:= ret;
end;

constructor TCyclopsDataElement.createWith(vGroupNumber, vElementNumber: word;
                    vVRC1, vVRC2: char; vValueLength: longword;
                    var vValue: TCyclopsByteArray);
begin
  GroupNumber:= vGroupNumber;
  ElementNumber:=  vElementNumber;
  VRC1:= vVRC1;
  VRC2:= vVRC2;
  ValueLength:= vValueLength;
  setLength(Value, ValueLength);
  Value:= copy(vValue, 0, length(vValue));
end;

function TCyclopsDataElement.tagString: string;
begin
  result:= '(' + TCyclopsGeneralFunctions.intToHexString(GroupNumber, 4) + ',' +
          TCyclopsGeneralFunctions.intToHexString(ElementNumber, 4) + ')';
end;

end.

//-----------------------------------------------------------------------------------------------------

unit uCyclopsDataElementsList;

interface

uses classes, uCyclopsDataElement;

type TCyclopsDataElementsList = class(TList)
  public
  procedure add(aDataElement: TCyclopsDataElement);
  procedure deleteAll;
  function getElement(groupNumber, elementNumber: integer):TCyclopsDataElement;
  function returnSubGroup(groupNumber: word): TCyclopsDataElementsList;
  function returnSubGroups(groupNumbers: array of word) : TCyclopsDataElementsList;
end;

implementation

{ TCyclopsDataElementsList }
//-------------------------------------------------------------------------
procedure TCyclopsDataElementsList.add(aDataElement: TCyclopsDataElement);
begin
  inherited add(aDataElement);
end;
//-------------------------------------------------------------------------
procedure TCyclopsDataElementsList.deleteAll;
```

```
var
  i: integer;
  element: TCyclopsDataElement;
begin
  for i:= 0 to Count- 1 do
    begin
    element:= items[i];
    element.Free;
    end;
end;
//----------------------------------------------------------------------------
function TCyclopsDataElementsList.getElement(groupNumber,
  elementNumber: integer): TCyclopsDataElement;
var
  i: integer;
  element, match: TCyclopsDataElement;
begin
  match:= nil;
  for i:= 0 to Count- 1 do
    begin
    element:= items[i];
    if (element.GroupNumber = groupNumber) and
      (element.elementNumber = elementNumber) then
      begin
      match:= element;
      break;
      end;
    end;
  result:= match;
end;
//----------------------------------------------------------------------------
function TCyclopsDataElementsList.returnSubGroup(
  groupNumber: word): TCyclopsDataElementsList;
var
  i: integer;
  element: TCyclopsDataElement;
  subGroup: TCyclopsDataElementsList;
begin
  subGroup:= TCyclopsDataElementsList.Create;
  for i:= 0 to Count- 1 do
    begin
    element:= items[i];
    if element.GroupNumber = groupNumber then
      subGroup.add(element);
    end;
  result:= subgroup;
end;
//----------------------------------------------------------------------------
function TCyclopsDataElementsList.returnSubGroups(
  groupNumbers: array of word): TCyclopsDataElementsList;
var
  cont : integer;
  i: integer;
  element: TCyclopsDataElement;
  subGroup: TCyclopsDataElementsList;
  groupNumber : word;
begin
  subGroup:= TCyclopsDataElementsList.Create;
  for cont := 0 to high(groupNumbers) do
  begin
    for i:= 0 to Count - 1 do
    begin
      element:= items[i];
      groupNumber := groupNumbers[cont];
      if (element.GroupNumber = groupNumber) then
        subGroup.add(element);
      end;
    end;
  result:= subgroup;
end;
```

```
//----------------------------------------------------------------------------
end.

//----------------------------------------------------------------------------------------------------

unit uCyclopsDCMReader;

interface

uses uCyclopsDICOMTransferSyntax, classes, uCyclopsDataElementsList,
     uCyclopsDICOMLETransferSyntax, uCyclopsDataElement,
     uCyclopsDICOMImplicitLETransferSyntax, uCyclopsDICOMBETransferSyntax,
     uCyclopsGeneralFunctions, sysUtils, dialogs,uCyclopsHandle;

type EDicomFileError = class (Exception);

type TCyclopsDCMReader = class
  protected
  Syntax: TCyclopsDICOMTransferSyntax;
  fDataElements: TCyclopsDataElementsList;
  ffileName: string;
  fastRead: boolean;
  public
  constructor createWithFile(vFileName: String; vFastRead: boolean);
  procedure finalizeInstance;

  procedure readFileMetaInformation;
  procedure getDataElements;
  function getImagePixelElement: TCyclopsDataElement;

  property DataElements: TCyclopsDataElementsList read fDataElements write fDataElements;
  property fileName: string read ffileName write ffileName;
end;
implementation

{ TCyclopsDCMReader }

constructor TCyclopsDCMReader.createWithFile(vFileName: String; vFastRead: boolean);
begin
  fileName:= vFileName;
  DataElements:= TCyclopsDataElementsList.Create;
  fastRead:= vFastRead;
end;

procedure TCyclopsDCMReader.finalizeInstance;
begin
  if Syntax <> nil then
    begin
    Syntax.finalizeInstance;
    Syntax.Free;
    end;
end;
//----------------------------------------------------------------------------
//take all data elements in .dcm file
procedure TCyclopsDCMReader.getDataElements;
var
  element: TCyclopsDataElement;
  syntaxUID: string;
  newSyntax: TCyclopsDICOMTransferSyntax;
begin
  Syntax:= TCyclopsDICOMLeTransferSyntax.createWithFile(fileName, fastRead);
  Syntax.skipFilePreamble;
  if not Syntax.checkDICOMPrefix then
    raise TCyclopsHandle.Create('Not a DICOM File!')
  else
    begin
    readFileMetaInformation;

    //now we must use the transfer Syntax indicated by
    //DE(0002,0010) for the rest of the file
```

```
    element:= DataElements.getElement($0002,$0010);
    syntaxUID:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
    CurrentTransFerSintax := syntaxUID;
    newSyntax:= nil;
    if syntaxUID = '1.2.840.10008.1.2' then  //Implicit VR, Little Endian
      newSyntax:= TCyclopsDICOMImplicitLETransferSyntax.continueFrom(Syntax)
    else
    if syntaxUID = '1.2.840.10008.1.2.1' then //Explicit VR, Little Endian
      newSyntax:= TCyclopsDICOMLETransferSyntax.continueFrom(Syntax)
    else
    if syntaxUID = '1.2.840.10008.1.2.2' then //Explicit VR, Big Endian
      newSyntax:= TCyclopsDICOMBETransferSyntax.continueFrom(Syntax);
    if newSyntax = nil then
      begin
      //throw exception, unknown syntax
      end
    else
      Syntax:= newSyntax;
    element:= Syntax.readDataElement;
    while element <> nil do
      begin
      DataElements.add(element);
      if (FastRead and (element.GroupNumber = $7FE0)) then
      //if it is set for fast read, and it has the last read DE
      //was of this group (probaly the pixel data length),
      //do not read the next DE, because it will countain pixel data
        break;
      element:= Syntax.readDataElement;
      end;
    end;
end;
//----------------------------------------------------------------------------
function TCyclopsDCMReader.getImagePixelElement: TCyclopsDataElement;
var
  element: TCyclopsDataElement;
  syntaxUID: string;
  newSyntax: TCyclopsDICOMTransferSyntax;
begin
  Syntax:= TCyclopsDICOMLeTransferSyntax.createWithFile(fileName, false);
  Syntax.skipFilePreamble;
  if not Syntax.checkDICOMPrefix then
    raise EDicomFileError.Create('Not a DICOM File!')
  else
    begin
    readFileMetaInformation;
    //now we must use the transfer Syntax indicated by
    //DE(0002,0010) for the rest of the file
    element:= DataElements.getElement($0002,$0010);
    syntaxUID:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
    newSyntax:= nil;
    if syntaxUID = '1.2.840.10008.1.2' then  //Implicit VR, Little Endian
      newSyntax:= TCyclopsDICOMImplicitLETransferSyntax.continueFrom(Syntax)
    else
    if syntaxUID = '1.2.840.10008.1.2.1' then //Explicit VR, Little Endian
      newSyntax:= TCyclopsDICOMLETransferSyntax.continueFrom(Syntax)
    else
    if syntaxUID = '1.2.840.10008.1.2.2' then //Explicit VR, Big Endian
      newSyntax:= TCyclopsDICOMBETransferSyntax.continueFrom(Syntax);
    if newSyntax = nil then
      begin
      //throw exception, unknown syntax
      end
    else
      Syntax:= newSyntax;
    element:= Syntax.readDataElement;
    if ((element.GroupNumber <> $7FE0) and
       (element.elementNumber <> $0010)) then
      repeat
        element:= Syntax.readDataElement;
      until ((element = nil) or
```

```
          ((element.GroupNumber = $7FE0) and
           (element.elementNumber = $0010)));
    end;
  result:= element;
end;
//----------------------------------------------------------------------------
procedure TCyclopsDCMReader.readFileMetaInformation;
var
  element: TCyclopsDataElement;
  groupLength, bytesRead: integer;
begin
  element:= Syntax.readDataElement;
  groupLength:= TCyclopsGeneralFunctions.infFromByteArrayLittleEndian(element.Value);
  bytesRead:= 0;
  repeat
    element:= Syntax.readDataElement;
    DataElements.add(element);
    bytesRead:= bytesRead + element.ValueLength + 4 + 2 + 4;
  until bytesRead >= groupLength;
end;

end.

//-------------------------------------------------------------------------------------------------------------

unit uCyclopsDICOMBETransferSyntax;

interface

uses uCyclopsDataElement, classes, sysUtils, uCyclopsGeneralFunctions,

  uCyclopsDICOMTransferSyntax, dialogs;

type TCyclopsDICOMBETransferSyntax = class(TCyclopsDICOMTransferSyntax)

  public

  procedure readFile;


  {Tag Functions}

  function readDataElement: TCyclopsDataElement; override;

end;

implementation

{ TCyclopsDICOMBETransferSyntax }

function TCyclopsDICOMBETransferSyntax.readDataElement: TCyclopsDataElement;

var

  GN, EN: word;

  VRC1, VRC2: char;

  b, value: TCyclopsByteArray;

  leng: longword;

  VR: string;

  bytesRead: integer;

begin
```

```
//CONTINUAR AQUI, transformar pra Big ENDIAN!!!!!!!!!!!!!!!!!!!!!!
  setLength(b, 4);
  bytesRead:= fileStream.Read(Pointer(b)^, 2);
  //This typecast Pointer() is necessary for read function proper work

  if bytesRead = 0 then
    begin
   result:= nil;
    exit;
    end;
  //This typecast Pointer() is necessary for read function proper work

  GN:= 0 or (b[1]) or (b[0] shl 8);   //follows little Endian Notation
  fileStream.Read(Pointer(b)^, 2);
  EN:= 0 or (b[1]) or (b[0] shl 8);   //follows little Endian Notation

  if (fastRead and (GN = $7FE0) and (EN = $0010)) then
    exit;

  fileStream.Read(Pointer(b)^, 2);
  VRC1:= char(b[0]);
  VRC2:= char(b[1]);
  vr:= '' + VRC1 + VRC2;
  if (vr = 'OB') or
     (vr = 'OW') or
     (vr = 'SQ') or
     (vr = 'OR') or
     (vr = 'UN') then

//DEs with these Explicit VRs have  the following format:

// --------------------------------------------------------------------------
// |GROUP No.|ELEM. No.|    VR   |RESERVED | LENGTH |      VALUE       |
// --------------------------------------------------------------------------
// | 2 bytes | 2 bytes | 2 bytes | 2 bytes | 4 bytes |    length bytes    |
```

```
// ------------------------------------------------------------------------


    begin
    fileStream.Read(Pointer(b)^, 2);  //skips reserved bytes of VR
    fileStream.Read(Pointer(b)^, 4);
    leng:= 0 or
          (b[3]) or
          (b[2] shl 8) or
         (b[1] shl 16) or
          (b[0] shl (23+1));
    end
   else
//DEs with other Explicit VRs have the following format:


// ------------------------------------------------------------------
// |GROUP No.|ELEM. No.|    VR   | LENGTH |   VALUE           |
// ------------------------------------------------------------------
// | 2 bytes | 2 bytes | 2 bytes | 2 bytes | length bytes      |
// ------------------------------------------------------------------


    begin
    fileStream.Read(Pointer(b)^, 2);
    leng:= 0 or
          (b[1]) or
          (b[0] shl 8);
    end;
   setLength(b, leng);
   fileStream.Read(Pointer(b)^, leng);
   value:= copy(b, 0, leng);
   result:= TCyclopsDataElement.createWith(GN, EN, VRC1, VRC2, leng, value);
end;

procedure TCyclopsDICOMBETransferSyntax.readFile;
begin
end;
end.

//-------------------------------------------------------------------------------------------------------------

unit UCyclopsDicomImage;
```

```
interface
        type TCyclopsDicomImage = class

private
   _generalImage          : string;
   _imagePlane                    : string;
   _imagePixel                    : string;
   _contrastBolus         : string;
   _overlayPlane          : string;
   _voilut                        : string;
   _sOPCommon                     : string;
   _series                        : string;
   _mrCachedImage         : string;
   _writeStream           : string;
   _minPixelValue         : string;
//   _maxPixelValue         : string;
protected

public

published
        property   generalImage        : string read _generalImage      write _generalImage;
   propertyimagePlane                  : string read _imagePlane        write _imagePlane;
   propertyimagePixel                  : string read _imagePixel        write _imagePixel;
   property   contrastBolus      : string read _contrastBolus write _contrastBolus;
   property   overlayPlane       : string read _overlayPlane write _overlayPlane;
   property   voilut                   : string read _voilut write _voilut;
   property   sOPCommon                : string read _sOPCommon write _sOPCommon;
   propertyseries                      : string read _series write _series;
   property   mrCachedImage      : string read _mrCachedImage write _mrCachedImage;
   property   writeStream              : string read _writeStream write _writeStream;
   property   minPixelValue      : string read _minPixelValue write _minPixelValue;

end;
implementation

end.

//-----------------------------------------------------------------------------------------------------

unit uCyclopsDICOMImplicitLETransferSyntax;

interface

uses uCyclopsDataElement, classes, sysUtils, uCyclopsGeneralFunctions,
  uCyclopsDICOMTransferSyntax, dialogs;

type TCyclopsDICOMImplicitLETransferSyntax = class(TCyclopsDICOMTransferSyntax)
  public
//  procedure readFile; override;
  function readDataElement: TCyclopsDataElement; override;
end;

implementation

{ TCyclopsDICOMImplicitLETransferSyntax }

function TCyclopsDICOMImplicitLETransferSyntax.readDataElement: TCyclopsDataElement;
var
  GN, EN: word;
  VRC1, VRC2: char;
  b, value: TCyclopsByteArray;  leng: longword;
  bytesRead: integer;
begin

//DEs with Implicit VRs have the following format:

// ----------------------------------------------------
// |GROUP No.|ELEM. No.| LENGTH |   VALUE          |
```

```
// ----------------------------------------------------
// | 2 bytes | 2 bytes | 4 bytes | length bytes      |
// ----------------------------------------------------

  setLength(b, 4);
  bytesRead:= fileStream.Read(Pointer(b)^, 2);
  //This typecast Pointer() is necessary for read function proper work

  if bytesRead = 0 then
    result:= nil
  else
    begin
    GN:= 0 or (b[0]) or (b[1] shl 8);   //follows Little Endian Notation
    fileStream.Read(Pointer(b)^, 2);
    EN:= 0 or (b[0]) or (b[1] shl 8);   //follows Little Endian Notation

    if (fastRead and (GN = $7FE0) and (EN = $0010)) then
      exit;

    VRC1:= #0; //Implicit VR
     VRC2:=#0;
    fileStream.Read(Pointer(b)^, 4);
    leng:= 0 or
         (b[0]) or
         (b[1] shl 8) or
        (b[2] shl 16) or
         (b[3] shl (23+1));
     setLength(value, leng);
    fileStream.Read(Pointer(value)^, leng);
    result:= TCyclopsDataElement.createWith(GN, EN, VRC1, VRC2, leng, value);
    end;
end;

end.

//-----------------------------------------------------------------------------------------------------------

unit uCyclopsDICOMLETransferSyntax;

interface

uses uCyclopsDataElement , classes, sysUtils, uCyclopsGeneralFunctions,
  uCyclopsDICOMTransferSyntax, dialogs;

type TCyclopsDICOMLETransferSyntax = class(TCyclopsDICOMTransferSyntax)
  public
//  procedure readFile; override;

  {Tag Functions}
  function readDataElement: TCyclo psDataElement; override;
end;

implementation

{ TCyclopsDICOMLETransferSyntax }

function TCyclopsDICOMLETransferSyntax.readDataElement: TCyclopsDataElement;
var
  GN, EN: word;
  VRC1, VRC2: char;
  b, value: TCyclopsByteArray;
  leng: longword;
  VR: string;
  bytesRead: integer;
begin
  setLength(b, 4);
  bytesRead:= fileStream.Read(Pointer(b)^, 2);
  //This typecast Pointer() is necessary for read function proper work

  if bytesRead = 0 then
```

```
  result:= nil
 else
  begin
  GN:= 0 or (b[0]) or (b[1] shl 8);   //follows little Endian Notation
  fileStream.Read(Pointer(b)^, 2);
  EN:= 0 or (b[0]) or (b[1] shl 8);   //follows little Endian Notation

  if (fastRead and (GN = $7FE0) and (EN = $0010)) then
    exit;

  fileStream.Read(Pointer(b)^, 2);
   VRC1:=char(b[0]);
  VRC2:= char(b[1]);
  vr:= '' + VRC1 + VRC2;
  if (vr = 'OB') or
    (vr = 'OW') or
    (vr = 'SQ') or
    (vr = 'OR') or
    (vr = 'UN') then

 //DEs with these Explicit VRs have the following format:

 // ---------------------------------------------------------------------
 // |GROUP No.|ELEM. No.|    VR   |RESERVED | LENGTH |      VALUE       |
 // ---------------------------------------------------------------------
 // | 2 bytes | 2 bytes | 2 bytes | 2 bytes | 4 bytes |   length bytes   |
 // ---------------------------------------------------------------------

    begin
    fileStream.Read(Pointer(b)^, 2);  //skips reserved bytes of VR
    fileStream.Read(Pointer(b)^, 4);
    leng:= 0 or
         (b[0]) or
         (b[1] shl 8) or
         (b[2] shl 16) or
         (b[3] shl (23+1));
    end
  else
 //DEs with other Explicit VRs have the following format:

 // ---------------------------------------------------------------
 // |GROUP No.|ELEM. No.|    VR   | LENGTH |   VALUE          |
 // ---------------------------------------------------------------
 // | 2 bytes | 2 bytes | 2 bytes | 2 bytes | length bytes     |
 // ---------------------------------------------------------------

    begin
    fileStream.Read(Pointer(b)^, 2);
    leng:= 0 or
         (b[0]) or
         (b[1] shl 8);
    end;
   setLength(value, leng);
   fileStream.Read(Pointer(value)^, leng);
   result:= TCyclopsDataElement.createWith(GN, EN, VRC1, VRC2, leng, value);
   end;
end;

end.

//-----------------------------------------------------------------------------------------------------------

unit uCyclopsDICOMServer;

interface

type TCyclopsDICOMServer = class

protected
  fHostName : String;
```

```
  fPort : String;
  fDicomDBServerName : String;
  fServerName : String;
public
  class function getcurrentServer: TCyclopsDICOMServer;
  constructor createWith(vHostName, vPort, vDicomDBServerName, vServerName: string);
  property HostName:string read fHostName write fHostName;
  property Port:string read fPort write fPort;
  property DicomDBServerName:string read fDicomDBServerName write fDicomDBServerName;
  property ServerName:string read fServerName write fServerName;
end;

var
  currentServer: TCyclopsDICOMServer;
implementation

{ TCyclopsDICOMServer }

constructor TCyclopsDICOMServer.createWith(vHostName, vPort,
  vDicomDBServerName, vServerName: string);
begin
  HostName:= vHostName;
  Port:= vPort;
  DicomDBServerName:= vDicomDBServerName;
  ServerName:= vServerName;
end;

class function TCyclopsDICOMServer.getcurrentServer: TCyclopsDICOMServer;
begin
  if currentServer = nil then
    currentServer:= TCyclopsDICOMServer.createWith('', '', '', '');
  result:= currentServer;
end;

end.

//----------------------------------------------------------------------------------------------------

unit uCyclopsDICOMTransferSyntax;

interface

uses uCyclopsDataElement, classes, sysUtils, uCyclopsGeneralFunctions;

type TCyclopsDICOMTransferSyntax = class
  protected
  fileName: string;
  fileStream: TFileStream;
  dataElements: TList;
  fastRead: boolean;
  public
  constructor createWithFile(vFileName: string; vFastRead: boolean);
  constructor continueFrom(anotherSyntax: TCyclopsDICOMTransferSyntax);
  procedure finalizeInstance;
  procedure skipFilePreamble;
  function checkDICOMPrefix: boolean;

  function readDataElement: TCyclopsDataElement; virtual; abstract;
end;

implementation

{ TCyclopsDICOMTransferSyntax }


function TCyclopsDICOMTransferSyntax.checkDICOMPrefix: boolean;
var
  buff: array [0..3] of char;
begin
  fileStream.Read(buff, 4);
```

```
  if (buff[0] = 'D') and
     (buff[1] = 'I') and
     (buff[2] = 'C') and
     (buff[3] = 'M') then
    result:= true
  else
    result:= false;
end;

constructor TCyclopsDICOMTransferSyntax.continueFrom(
  anotherSyntax: TCyclopsDICOMTransferSyntax);
begin
  fileName:= anotherSyntax.FileName;
  fileStream:= anotherSyntax.fileStream;
  dataElements:= anotherSyntax.dataElements;
  fastRead:= anotherSyntax.fastRead;
end;

constructor TCyclopsDICOMTransferSyntax.createWithFile(vFileName: string; vFastRead: boolean);
begin
  fileName:= vFileName;
  FileStream:= TFileStream.Create(vFileName, fmOpenRead);
  DataElements:= TList.Create;
  fastRead:= vFastRead;
end;

procedure TCyclopsDICOMTransferSyntax.finalizeInstance;
begin
  FileStream.free;
end;

procedure TCyclopsDICOMTransferSyntax.skipFilePreamble;
var
  buff: array [0..127] of char;
begin
  fileStream.ReadBuffer(buff, 128);
end;

end.

//----------------------------------------------------------------------------------------------------

unit uCyclopsFileManager;

interface

uses sysUtils, classes, forms, uCyclopsDCMReader, uCyclopsDataElement,
     uCyclopsDataElementsList, uCyclopsPatient, uCyclopsStudy, uCyclopsSeries,
     uCyclopsGeneralFunctions, uCyclopsImage, uCyclopsTempImagesManager;

type TCyclopsFileManager = class
  protected
  DCMFileReader: TCyclopsDCMReader;
  FPatientsList: TList;
  FTempImageManager: TCyclopsTempIma gesManager;
  public
  constructor create;

  function openDCMFile(filename: string; fastRead: boolean): TCyclopsImage;
  function openDCMFileStudyInfo(filename: string): TCyclopsImage;
  function addPatientWithDEs(dataElements: TCyclopsDataElementsList; var alreadyExists: boolean): TCyclopsPatient;
  function addStudyWithDEsToPatient(dataElements: TCyclopsDataElementsList; patient: TCyclopsPatient): TCyclopsStudy;
  function addSerieWithDEsToStudy(dataElements: TCyclopsDataElementsList; study: TCyclopsStudy): TCyclopsSeries;
  function addImageWithDEsToSerie(dataElements: TCyclopsDataElementsList; serie: TCyclopsSeries; withPixelData: boolean):
TCyclopsImage;

  function patientWithID(ID: string): TCyclopsPatient;
  function StudyWithIDOnPatient(ID: string; patient: TCyclopsPatient): TCyclopsStudy;
  function SerieWithIDOnStudy(ID: string; study: TCyclopsStudy): TCyclopsSeries;
  function ImageWithIDOnSerie(ID: string; serie: TCyclopsSeries): TCyclopsImage;
```

```
  property PatientsList : TList read FPatientsList  write FPatientsList;
  property TempImageManager: TCyclopsTempImagesManager read FTempImageManager write FTempImageManager;
end;

implementation

{ TCyclopsFileManager }
//---------------------------------------------------------------------------
function TCyclopsFileManager.addImageWithDEsToSerie(
  dataElements: TCyclopsDataElementsList;
  serie: TCyclopsSeries;
  withPixelData: boolean): TCyclopsImage;
var
  i: integer;
  element: TCyclopsDataElement;
  image: TCyclopsImage;
begin
  element:= dataElements.getElement($0008, $0018);
  //search if this image already exists in the serie
  image := ImageWithIDOnSerie(TCyclopsGeneralFunctions.byteArrayAsString(element.Value), serie);

  if image = nil then
  begin
    //generate study data
    image := TCyclopsImage.createWith(nil, TempImageManager, withPixelData);
    image.Series:= serie;
    for i:= 0 to dataElements.Count - 1 do
    begin
      element:= dataElements.items[i];
      image.processDataElement(element);
    end;
    serie.images.Add(image);
  end;
  result:= image;
end;
//---------------------------------------------------------------------------
function TCyclopsFileManager.addPatientWithDEs(
  dataElements: TCyclopsDataElementsList; var alreadyExists: boolean): TCyclopsPatient;
var
  i: integer;
  element: TCyclopsDataElement;
  patient: TCyclopsPatient;
begin
  element:= dataElements.getElement($0010, $0020);
  patient:= patientWithID(TCyclopsGeneralFunctions.byteArrayAsString(element.Value));
  if patient = nil then
    begin
    alreadyExists:= false;
    patient:= TCyclopsPatient.create;
    for i:= 0 to dataElements.Count - 1 do
      begin
      element:= dataElements.items[i];
      patient.processDataElement(element);
      end;
      PatientsList.Add(patient);
    end
  else
    alreadyExists:= true;
  result:= patient;
end;
//---------------------------------------------------------------------------
function TCyclopsFileManager.addSerieWithDEsToStudy(
  dataElements: TCyclopsDataElementsList;
  study: TCyclopsStudy): TCyclopsSeries;
var
  i: integer;
  element: TCyclopsDataElement;
  serie: TCyclopsSeries;
begin
```

```
  element:= dataElements.getElement($0020, $000E);
  //search if this study already exists in any patient
  serie := SerieWithIDOnStudy(TCyclopsGeneralFunctions.byteArrayAsString(element.Value), study);

  if serie = nil then
  begin
    //generate study data
    serie := TCyclopsSeries.create;
    serie.study:= study;
    for i:= 0 to dataElements.Count - 1 do
    begin
      element:= dataElements.items[i];
      serie.processDataElement(element);
    end;
    study.series.Add(serie);
  end;
  result:= serie;
end;
//----------------------------------------------------------------------------
function TCyclopsFileManager.addStudyWithDEsToPatient(
              dataElements: TCyclopsDataElementsList; patient : TCyclopsPatient): TCyclopsStudy;
var
  i: integer;
  element: TCyclopsDataElement;
  study: TCyclopsStudy;
begin
  element:= dataElements.getElement($0020, $0010);
  //search if this study already exists in any patient
  study := StudyWithIDOnPatient(TCyclopsGeneralFunctions.byteArrayAsString(element.Value), patient);

  if study = nil then
  begin
    //generate study data
    study := TCyclopsStudy.create;
    study.patient:= patient;
    for i:= 0 to dataElements.Count - 1 do
    begin
      element:= dataElements.items[i];
      study.processDataElement(element);
    end;
              patient.studies.Add(study);
  end;
  result:= study;
end;
//----------------------------------------------------------------------------
constructor TCyclopsFileManager.create;
begin
  PatientsList := TList.Create;
  TempImageManager:= TCyclopsTempImagesManager.createOn(ExtractFilePath(Application.ExeName) + 'tmp');
end;
//----------------------------------------------------------------------------
function TCyclopsFileManager.ImageWithIDOnSerie(ID: string;
  serie: TCyclopsSeries): TCyclopsImage;
var
  i: integer;
  image, match          : TCyclopsImage;
begin
  match:= nil;
  for i:= 0 to serie.images.Count- 1 do
    begin
    image := serie.images.Items[i];
    if image .SOPInstanceUID = ID then
      begin
      match:= image;
      break;
              end;
    end;
  result:= match;
end;
//----------------------------------------------------------------------------
```

```
function TCyclopsFileManager.openDCMFile(filename: string; fastRead: boolean): TCyclopsImage;
var
  group   : TCyclopsDataElementsList;
  patient : TCyclopsPatient;
  study   : TCyclopsStudy;
  series  : TCyclopsSeries;
  image       : TCyclopsImage;
  patientExists: boolean;
begin
  DCMFileReader:= TCyclo psDCMReader.createWithFile(fileName, fastRead);
  DCMFileReader.getDataElements;
  //generate group of patients tags

  group:= DCMFileReader.DataElements.returnSubGroup($0010);
  patient := addPatientWithDEs(group, patientExists);
  patient.Files.Add(fileName);

  //generate group of studies tags

  group:= DCMFileReader.DataElements.returnSubGroups([$0008, $0010, $0020]);
  study := addStudyWithDEsToPatient(group,patient);
  study.Files.Add(fileName);

  //generate group of series tags

  group:= DCMFileR eader.DataElements.returnSubGroups([$0020, $0008, $0018]);
  series := addSerieWithDEsToStudy(group,study);
  series.Files.Add(fileName);

  //generate group of images tags

  group:= DCMFileReader.DataElements.returnSubGroups([$0008, $0020, $0028, $7FE0]);
  image:= addImageWithDEsToSerie(group,series, fastRead);
  image.DataElements:= DCMFileReader.DataElements;
  image.FileName:= fileName;
  image.deleteDataElements;

  //close the file
  DCMFileReader.finalizeInstance;
  DCMFileReader.Free;

  result:= image;
end;
//---------------------------------------------------------------------------
function TCyclopsFileManager.openDCMFileStudyInfo(
  filename: string): TCyclopsImage;
var
  group   : TCyclopsDataElementsList;
  patient : TCyclopsPatient;
  study   : TCyclopsStudy;
//  series  : TCyclopsSeries;
  image       : TCyclopsImage;
  patientExists: boolean;
begin
  DCMFileReader:= TCyclopsDCMReader.createWithFile(fileName, true);
  DCMFileReader.getDataElements;
  //generate group of patients tags

  group := DCMFileReader.DataElements.returnSubGroup($0010);
  patient := addPatientWithDEs(group, patientExists);
  patient.Files.Add(fileName);

  //generate group of studies tags

  group:= DCMFileReader.DataElements.returnSubGroups([$0008, $0010, $0020]);
  study := addStudyWithDEsToPatient(group,patient);
  study.Files.Add(fileName);

  //generate group of series tags
```

```
  DCMFileReader.finalizeInstance;
  DCMFileReader.Free;

  result:= image;
end;
//----------------------------------------------------------------------------
function TCyclopsFileManager.patientWithID(ID: string): TCyclopsPatient;
var
  i: integer;
  patient, match: TCyclopsPatient;
begin
  match:= nil;
  for i:=0 to PatientsList.Count - 1 do
    begin
    patient:= PatientsList.items[i];
     if patient.data.patientID = ID then
       begin
      match:= patient;
       break;
       end;
    end;
  result:= match;
end;
//----------------------------------------------------------------------------
function TCyclopsFileManager.SerieWithIDOnStudy(ID: string;
  study: TCyclopsStudy): TCyclopsSeries;
var
  i: integer;
  Serie, match          : TCyclopsSeries;
begin
  match:= nil;
  for i:= 0 to study.series.Count- 1 do
    begin
    serie := study.series.Items[i];
    if serie.generalSeries.seriesInstanceUID= ID then
      begin
      match:= serie;
      break;
                end;
    end;
  result:= match;
end;
//----------------------------------------------------------------------------
function TCyclopsFileManager.StudyWithIDOnPatient(ID: string; patient: TCyclopsPatient): TCyclopsStudy;
var
  i: integer;
  Study, match          : TCyclopsStudy;
begin
  match:= nil;
  for i:= 0 to patient.studies.Count- 1 do
    begin
    study := patient.studies.Items[i];
    if study.generalStudy.studyID = ID then
      begin
      match:= study;
      break;
              end;
    end;
  result:= match;
end;
//----------------------------------------------------------------------------
end.

//----------------------------------------------------------------------------------------------------------

unit uCyclopsGeneralFunctions;

interface

uses Windows;
```

```
type TCyclopsByteArray = array of byte;

type TCyclopsGeneralFunctions = class

  public
    class procedure initialize;
    class function intToHexString(value: integer; digitCount: integer):string;
    class function intFromByteArray(byteArray: array of byte; start, stop: integer):integer;
    class function byteArrayAsString(byteArray: array of byte):string;
    class function infFromByteArrayLittleEndian(byteArray: array of byte): integer;
    class function infFromByteArray(byteArray: array of byte): integer;
    class function copyCyclopsByteArray(var source: TCyclopsByteArray): TCyclopsByteArray;
  end;

var
  hexChars: array [0..15] of char;
  //this is suposed to be a class property of TCyclopsGeneralFunctions
  CurrentTransFerSintax                    : string;
implementation
{ TCyclopsGeneralFunctions }
//---------------------------------------------------------------------------
class function TCyclopsGeneralFunctions.byteArrayAsString(
  byteArray: array of byte): string;
var
  i, leng: integer;
  str: string;
begin
  leng:= length(byteArray);
  if leng > 0 then
    begin
    setLength(str, leng);
    for i:= 0 to leng - 1 do
      str[i + 1]:= char(byteArray[i]);

    //is the string null terminated?
    if str[leng] = #0 then
      //we don´t want null terminated strings, so remove the null char
      setLength(str, leng - 1);
    result:= str;
    end
  else
    result:= '';
end;
//---------------------------------------------------------------------------
class function TCyclopsGeneralFunctions.copyCyclopsByteArray(
  var source: TCyclopsByteArray): TCyclopsByteArray;
var
  dest: TCyclopsByteArray;
  leng: integer;
begin
  leng:= length(source);
  setLength(dest, leng);
  copyMemory(dest, source, leng);
  result:= dest;
end;
//---------------------------------------------------------------------------
class function TCyclopsGeneralFunctions.infFromByteArray(
  byteArray: array of byte): integer;
var
  i, value: integer;
begin
            if CurrentTransFerSintax = '1.2.840.10008.1.2.2' then   //big endian
    begin
      value:= 0;
      for i := high(byteArray) downto 0 do
                value:= value or (byteArray[i] shl ((high(byteArray) - i)*8));
      result:= value;
    end else
            //little endian
```

```
    begin
      value:= 0;
      for i:= 0 to high(byteArray) do
                value:= value or (byteArray[i] shl (i*8));
      result:= value;
    end;
end;
//----------------------------------------------------------------------------
class function TCyclopsGeneralFunctions.infFromByteArrayLittleEndian(
  byteArray: array of byte): integer;
var
  i, value: integer;
begin
    value:= 0;
    for i:= 0 to high(byteArray) do
      value:= value or (byteArray[i] shl (i*8));
    result:= value;
end;
//----------------------------------------------------------------------------
class procedure TCyclopsGeneralFunctions.initialize;
begin
  hexChars[0 ]:= '0';
  hexChars[1 ]:= '1';
  hexChars[2 ]:= '2';
  hexChars[3 ]:= '3';
  hexChars[4 ]:= '4';
  hexChars[5 ]:= '5';
  hexChars[6 ]:= '6';
  hexChars[7 ]:= '7';
  hexChars[8 ]:= '8';
  hexChars[9 ]:= '9';
  hexChars[10]:= 'A';
  hexChars[11]:= 'B';
  hexChars[12]:= 'C';
  hexChars[13]:= 'D';
  hexChars[14]:= 'E';
  hexChars[15]:= 'F';
end;


//----------------------------------------------------------------------------
class function TCyclopsGeneralFunctions.intFromByteArray(
  byteArray: array of byte; start, stop: integer): integer;
var
  i, value: integer;
begin
  value:= 0;
  for i:= 0 to stop - start do
    value:= value or (bytearray[i] shl i * 8);
  result:= value;
end;

//----------------------------------------------------------------------------
class function TCyclopsGeneralFunctions.intToHexString(value,
  digitCount: integer): string;
var
  i: integer;
  str: string;
  dig: byte;
begin
  str:= '';
  for i:=1 to digitCount do
     begin
     dig:= value and $F;
     str:= hexChars[dig] + str;
     value:= value shr 4;
     end;
  result:= str;
end;
//----------------------------------------------------------------------------
```

```
end.

//----------------------------------------------------------------------------------------------------

unit uCyclopsGeneralSeriesModule;

//----------------------------------------------------------------
//  This is a Smalltalk class converted with
//  the ToDelphiConversor, created by Charles I. Wust,
//  member of The Cyclops Project.
//----------------------------------------------------------------

interface

type TCyclopsGeneralSeriesModule = class

protected
  FbodyPartExamined : string;
  FlargestPixelValueInSeries : string;
  Fmodality : string;
  FoperatorsName : string;
  FpatientPosition: string;
  FperformingPhysiciansName: string;
  FprotocolName : string;
  FseriesDate : string;
  FseriesDescription : string;
  FseriesInstanceUID : string;
  FseriesNumber : string;
  FseriesTime : string;
  FsmallestPixelValueInSeries : string;
public
  property bodyPartExamined : string read FbodyPartExamined write FbodyPartExamined;
  property largestPixelValueInSeries : string read FlargestPixelValueInSeries write FlargestPixelValueInSeries;
  property modality : string read Fmodality write Fmodality;
  property operatorsName : string read FoperatorsName write FoperatorsName;
  property patientPosition : string read FpatientPosition write FpatientPosition;
  property performingPhysiciansName : string read FperformingPhysiciansName write FperformingPhysiciansName;
  property protocolName : string read FprotocolName write FprotocolName;
  property seriesDate : string read FseriesDate write FseriesDate;
  property seriesDescription : string read FseriesDescription write FseriesDescription;
  property seriesInstanceUID : string read FseriesInstanceUID write FseriesInstanceUID;
  property seriesNumber : string read FseriesNumber write FseriesNumber;
  property seriesTime : string read FseriesTime write FseriesTime;
  property smallestPixelValueInSeries : string read FsmallestPixelValueInSeries write FsmallestPixelValueInSeries;
end;

implementation

{TCyclopsGeneralSeriesModule}

end.

//----------------------------------------------------------------------------------------------------

unit uCyclopsGeneralStudyModule;

//----------------------------------------------------------------
//  This is a Smalltalk class converted with
//  the ToDelphiConversor, created by Charles I. Wust,
//  member of The Cyclops Project.
//----------------------------------------------------------------

interface

type TCyclopsGeneralStudyModule = class

protected
  FaccessionNumber : string;
  FnameOfPhysicianReadingStudy: string;
```

```
  FreferringPhysiciansName : string;
  FstudyDate : string;
  FstudyDescription : string;
  FstudyID : string;
  FstudyInstanceUID : string;
  FstudyTime : string;
public
  property accessionNumber : string read FaccessionNumber write FaccessionNumber;
  property nameOfPhysicianReadingStudy : string read FnameOfPhysicianReadingStudy write FnameOfPhysicianReadingStudy;
  property referringPhysiciansName : string read FreferringPhysiciansName write FreferringPhysiciansName;
  property studyDate : string read FstudyDate write FstudyDate;
  property studyDescription : string read FstudyDescription write FstudyDescription;
  property studyID : string read FstudyID write FstudyID;
  property studyInstanceUID : string read FstudyInstanceUID write FstudyInstanceUID;
  property studyTime : string read FstudyTime write FstudyTime;
end;

implementation

{TCyclopsGeneralStudyModule}

end.

//-------------------------------------------------------------------------------------------------------

//description:
//
//
//author: Daniel Duarte Abdala
//
//Date:                 16/04/2002
//
unit uCyclopsHandle;

interface
uses
        classes,sysUtils;

type TCyclopsHandle = class (Exception)

protected
        Filename  : string;
        LogFile           : TextFile;
   owner          : TComponent;
   fMethodName : String;
   flogFile : string;
public

   constructor CreateWith(AOwner: TComponent; logFName : string);
        constructor CreateSimple(aString : string);
   destructor        Destroy;
   procedure         logError;
        procedure showError;
published
        property methodName : string   read      fMethodName
                                                                write  fMethodName;
        property logFileName: string    read      flogFile
                                                                write  flogFile;

end;
implementation

{ TCyclopsHandle }
//---------------------------------------------------------------------------
constructor TCyclopsHandle.CreateSimple(aString: string);
begin
   Self.Message := aString;
end;
//---------------------------------------------------------------------------
constructor TCyclopsHandle.CreateWith(AOwner: TComponent; logFName : string);
```

```
begin
   owner := AOwner;
   flogFile := logFName;
end;
//---------------------------------------------------------------------------
destructor TCyclopsHandle.Destroy;
begin

end;
//---------------------------------------------------------------------------
procedure TCyclopsHandle.logError;
begin
   // prepares log Filename
    Filename := ChangeFileExt (flogFile, '.log');
   AssignFile (LogFile, Filename);
   if FileExists (FileName) then
            Append (LogFile) // open existing file
   else
            Rewrite (LogFile); // create a new one
   // write to the file and show error
   Writeln (LogFile, DateTimeToStr (Now) + ':' + self.message);
            // close the file
            CloseFile (LogFile);
end;
//---------------------------------------------------------------------------
//description:          Display a dialog box with a error description
procedure TCyclopsHandle.showError;
begin

end;
//---------------------------------------------------------------------------
end.
//---------------------------------------------------------------------------------------------------------
unit UCyclopsIEDefinitions;

interface

        Type    Sex                     = (Male, Female, Other);
      Validation  = (YES, NO, UNKNOWM);

      tyAE                  =       String; //Application Entity nmaximun 16 bytes
                            tyAS                    =   String; //4 bytes fixed                   (PS3.5 pg 21)
      tyAT                  =       String; //4 bytes fixed           (PS3.5 pg 21)
       tyCS     =   String;     //maximun16 bytes

implementation

end.
//---------------------------------------------------------------------------------------------------------

(*****************************************************************************
CLASS   : TCyclopsImageList
Author     :   Daniel Duarte Abdala (Caju)
                            Charles Wust

Date       :        09/10/2001

Description          : This class is a abstraction of a DICOM image.


Atributes :
   pOriginalWidth               : a integer representing the original image width  |(normaly are the same)
   pOriginalHeight              : a integer representing the original image height |
   pImageFileTipe     : especify the image source type
   pCurrentOrientation          : specify the actual orientation
   fSOPInstanceUID     : unique identifier of image   //this should be placed in a sub-module
   fpixelValue       : array of small integer wit h the original HU values of image
   fWindowCenter       : integer representing the window center
   fWindowWidth        : integer representing the window width
```

```
   ftempIndex        : temporary image index for swap on disk
   ftempImageManager          : ref to image temp manager object
   fImageNumber               : image number on serie


Methods :
   name : function get(Index1, Index2: integer): SmallInt;
   description: returns the HU value at index

   name : procedure put(Index1, Index2: integer; item: SmallInt);
   description :

   name : procedure getHUValuesFromDataElement(element: TCyclopsDataElement; var targetArray: array of SmallInt);
   description :

   name : function getBitmap: TBitmap;
   description :

          name : constructor CreateWith(AOwner: TComponent; tempImgManager: TCyclopsTempImagesManager);
   description :

   name : procedure processDataElement(element: TCyclopsDataElement);
   description :

   name : function getOriginalBitmap: Graphics.TBitmap;
   description :

   name : property  pixelValue[Index1: integer; Index2: integer] : SmallInt read get write put;
   description :


Related Errors        :


Need to by Done       :


*******************************************************************************)
unit UCyclopsImage;

interface
uses
          Windows, ExtCtrls,Dialogs, Classes, uCyclopsDataElement, uCyclopsGeneralFunctions, graphics,
   sysUtils, uCyclopsTempImagesManager, uCyclopsSeries, Printers, types, uCyclopsDataElementsList,
   uCyclopsDCMReader, Forms;
const
  annonymous = false;

type
          FileType   = (ftBmpFile, ftJpgFile);
   OrientationType    = (otTop, otRight, otBottom, otLeft);

          TCyclopsImage = class (TImage)

private
   converted          : boolean;
   pOriginalWidth             : integer;
   pOriginalHeight            : integer;
   pImageFileTip e      : FileType;
   pCurrentOrientation        : OrientationType;
   fSOPInstanceUID      : string;   //this should be placed in a sub-module

   fWindowCenter       : integer;
   fWindowWidth        : integer;
   fOriginalWindowCenter: integer;
   fOriginalWindowWidth : integer;
   ftempIndex          : integer;
   ftempImageManager          : TCyclopsTempImagesManager;
   fImageNumber               : integer;
   fSeries            : TCyclopsSeries;
```

```
    fSliceThickness                 : Integer;  //this number is in milimethers
    fSamplesPerPixel    : Integer;
    fDataElements       : TCyclopsDataElementsList;
    fFileName           : string;

//    MinHUValue                    : integer;
//    MaxHUValue                    : integer;
//          RGBValuesOriginal   : array of byte;
    pal: PLogPalette;

    function get(Index1, Index2: integer): SmallInt;
    procedure       put(Index1, Index2: integer; item: SmallInt);
    procedure       getHUValuesFromDataElement(element: TCyclopsDataElement; var targetArray: array of SmallInt);
    procedure       getUSValuesFromDataElement(element: TCyclopsDataElement; var targetArray: array of SmallInt);
    procedure       processImageInfoElement(element: TCyclopsDataElement);
    procedure       setImageNumber(number: integer);

    function calcAllImageHUValues: TBitmap;
public
    fpixelValue         : array of SmallInt;

    constructor CreateWith(AOwner: TComponent; tempImgManager: TCyclopsTempImagesManager; withPixelData: boolean);
    destructor      Destroy;  override;
    procedure       processDataElement(element: TCyclopsDataElement);
    function        getOriginalBitmap: TBitmap;
    function        getOriginalBitmapFromHUValues: TBitmap;
    function        getOriginalBitmapFromUSValues: TBitmap;
    function        isConverted: boolean;
    procedure       getOriginalWindowValues;
    procedure       print;
    procedure       setNewWindowValues(newWindowCenter, NewWindowWidth:integer);
    procedure       recoverPixelInformation;
    procedure       deleteDataElements;
    property        pixelValue[Index1: integer; Index2: integer] : SmallInt read get write put;
    property        DataElements: TCyclopsDataElementsList read fDataElements write fDataElements;
    function getBitmap: TBitmap;
    procedure       copy(tmpImg : tcyclopsImage);
    //
    procedure       changePalette(newWindowCenter,NewWindowWidth: integer);

published
        property    originalWidth           : Integer   read pOriginalWidth
                                                                                write
pOriginalWidth;
        property    OriginalHeight          : Integer   read pOriginalHeight
                                                                                write
pOriginalHeight;
        property    ImageFileTipe           : FileType  read pImageFileTipe
                                                                                write
pImageFileTipe;
        property    CurrentOrientation      : OrientationType
                                                                                read
pCurrentOrientation
                                                                                write
pCurrentOrientation;
        property    SOPInstanceUID          : string    read fSOPInstanceUID
                                                                        write fSOPInstanceUID;
        property    WindowCenter            : integer   read fWindowCenter
                                                                                write
fWindowCenter;
        property    WindowWidth             : integer   read fWindowWidth
                                                                                write
fWindowWidth;
        property    OriginalWindowCenter: integer
                                                                        read fOriginalWindowCenter
                        write fOriginalWindowCenter;
        property    OriginalWindowWidth     : integer   read fOriginalWindowWidth
                                                                                write
fOriginalWindowWidth;
    property        Bitmap                          :TBitmap        read getBitmap;
```

```
   property   tempIndex                              : integer   read ftempIndex
                                write ftempIndex;
   property   tempImageManager          : TCyclopsTempImagesManager
                                                                                read
ftempImageManager
                                write ftempImageManager;
   property   imageNumber                           : integer    read fimageNumber
                                                                                write
setImageNumber;

   propertySliceThickness               : Integer    read fSliceThickness
                                                                                write
fSliceThickness;
   propertySeries                 : TCyclopsSeries      read fSeries
                                                                                write
fSeries;
   propertySamplesPerPixel      : Integer    read fSamplesPerPixel
                                                                                write
fSamplesPerPixel;
   propertyFileName                         : string     read fFileName
                                                                                write
fFileName;
end;

implementation

uses Controls;

{ TCyclopsImage }

//---------------------------------------------------------------------------
constructor TCyclopsImage.CreateWith(AOwner: TComponent;
  tempImgManager: TCyclopsTempImagesManager; withPixelData: boolean);
begin
  inherited create(AOwner);
  Picture.Bitmap:= nil;
  tempImageManager:= tempImgManager;
  converted:= not withPixelData;
  fileName:= '';
  tempIndex:= -1;
end;
//---------------------------------------------------------------------------
//description: return a pixel value in HU values
function TCyclopsImage.get(Index1, Index2: integer): SmallInt;
begin
  result:= fpixelValue[Index1 * originalWidth + Index2];
end;
//---------------------------------------------------------------------------
//description: set a pixel value in HU values
procedure TCyclopsImage.put(Index1, Index2: integer; item: SmallInt);
begin
  fpixelValue[Index1 *originalWidth + Index2]:= item;
end;
//---------------------------------------------------------------------------
//description: returns a original temporary bitmap from temp files
function TCyclopsImage.getBitmap: TBitmap;
begin
  result:= tempImageManager.returnBitmap(tempIndex);
end;
//---------------------------------------------------------------------------
//description: responsible to parse DCM file and retreave all HU values
procedure TCyclopsImage.getHUValuesFromDataElement(
  element: TCyclopsDataElement; var targetArray: array of SmallInt);
var
  bAlloc, i, j: integer;
  v: word;
  b: byte;
  p: PInteger;
  HUValue: SmallInt;
begin
```

```
    //this works for Little Endian Byte Ordering
    bAlloc:= 2;
    setLength(fpixelValue, length(element.value) div bAlloc);
    for i:= 0 to (length(element.value) div bAlloc)- 1 do
    begin
       v:= 0;
       for j:=0 to bAlloc  - 1 do
       begin
          if CurrentTransFerSintax = '1.2.840.10008.1.2.2' then        //read as big endian
          begin
                      b:= element.value[(i * bAlloc) + j];
            v := v or (b shl (8 * (balloc - 1 - j)));
          end else
              //read as little endian
          begin
                       b:= element.value[(i * bAlloc) + j];
                      v:= v or (b shl (8 * j));
          end;
                end;
       p:= @v;
       HUValue:= p^;
       fpixelValue[i]:= HUValue;
    end;
end;
//------------------------------------------------------------------------------
//description:
function TCyclopsImage.getOriginalBitmap: TBitmap;
var
  x,y: integer;
  fbmp: TBitmap;
begin
  if converted then
    begin
    if Self.Series.generalSeries.modality = 'US' then
     fbmp:= getOriginalBitmapFromUSValues
    else
     fbmp:= getOriginalBitmapFromHUValues;

    if annonymous then
              for y:= 0 to 39 do
       for x:= 0 to fbmp.Width - 1 do
                          fbmp.Canvas.Pixels[y,x]:= 0;
    end       else
    begin
              fbmp:= nil;
    end;

  result:= fbmp;
end;
//------------------------------------------------------------------------------
//description: process all data elements to construct a DICOM image.
procedure TCyclopsImage.processDataElement(
  element: TCyclopsDataElement);
begin
  case element.GroupNumber of
    $0008:
     case element.ElementNumber of
       0000:
     end;
    $0020:
     case element.ElementNumber of
       $0013: imageNumber := strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
     end;
    $0028:
     case element.ElementNumber of
       $0002:SamplesPerPixel:= TCyclopsGeneralFunctions.infFromByteArray(element.Value);
       $0010:originalHeight:= TCyclopsGeneralFunctions.infFromByteArray(element.Va lue);
       $0011:originalWidth:= TCyclopsGeneralFunctions.infFromByteArray(element.Value);
       $1050:begin
            WindowCenter:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
```

```
            OriginalWindowCenter:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            end;
        $1051:begin
            WindowWidth:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            OriginalWindowWidth:= strtoint(trim(TCyclopsGeneralFunctions.byteArrayAsString(element.value)));
            end;
      end;
    $7FE0:
      case element.ElementNumber of
        $0010: processImageInfoElement(element);
      end;
  end;
end;
//-----------------------------------------------------------------------------
//description: set the current window width and Center to original values
procedure TCyclopsImage.getOriginalWindowValues;
begin
  windowCenter:= OriginalWindowCenter;
  if windowCenter = 0 then
              windowCenter:= 1791; //default value
  windowWidth:= OriginalWindowWidth;
  if windowWidth = 0 then
              windowWidth := 3583; //default value
end;
//-----------------------------------------------------------------------------
//description: used to take Ultra-Sonography data
procedure TCyclopsImage.getUSValuesFromDataElement(
  element: TCyclopsDataElement; var targetArray: array of SmallInt);
var
  i: integer;
begin
  setLength(fPixelValue, length(element.value));
  for i:= 0 to high(element.value) do
    fpixelValue[i]:= element.value[i];
end;
//-----------------------------------------------------------------------------
//description:

function TCyclopsImage.getOriginalBitmapFromHUValues: TBitmap;
begin
              result := calcAllImageHUValues;
end;
//-----------------------------------------------------------------------------
function TCyclopsImage.getOriginalBitmapFromUSValues: TBitmap;
var
  fbmp: TBitmap;
  i, j, jDown: integer;
  v1, v2, v3, gIndex, bIndex: integer;
begin
  fbmp:= TBitmap.Create;
//  fbmp.SetSize(originalWidth, OriginalHeight);
  fbmp.Width          :=  originalWidth;
  fbmp.Height         :=  OriginalHeight;
  jDown:= originalHeight - 1;
  for j:= 0 to originalHeight - 1 do
    begin
    for i:= 0 to originalWidth - 1 do
      begin
      v1:= fpixelValue[(j * 640) + i];
      if SamplesPerPixel = 1 then
        begin
        v2:= v1;
        v3:= v1;
        end
      else
        begin
        gIndex:= ((originalHeight - 1) * 640) + originalWidth;
        bIndex:= gIndex * 2;
        v2:= fpixelValue[gIndex + (j * 640) + i];
        v3:= fpixelValue[bIndex + (j * 640) + i];
```

```
      end;
     fbmp.Canvas.Pixels[i,j] := v1 or (v2 shl 8) or (v3 shl 16);//FRGB(v1, v2, v3);
     end;
    dec(jDown);
    end;
  result:= fbmp;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImage.print;
var
  patientName: string;
  topMargin, LeftMargin, RightMargin: integer;
  imgRect: TRect;
  bmp: TBitmap;
begin
  bmp:= getBitmap;
  patientName:= Series.study.patient.data.patientsName;
  Leftmargin:= trunc(printer.PageWidth * 0.2);
  RightMargin:= trunc(printer.PageWidth * 0.9);
  topMargin:= trunc(printer.PageHeight * 0.1);
  printer.BeginDoc;
  printer.Canvas.Font.Name:= 'Arial';
  printer.Canvas.TextOut(LeftMargin, topMargin, 'Pat ient Name: ' + patientName);
  imgRect:= Rect(LeftMargin, TopMargin + 300, RightMargin, trunc(((RightMargin - LeftMargin) / bmp.Width) * bmp.Height) +
300);
  printer.Canvas.StretchDraw(imgRect, bmp);
  printer.EndDoc;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImage.setNewWindowValues(newWindowCenter, NewWindowWidth:integer);
var
  bmp: Graphics.TBitmap;
  i, j, min, max: integer;
  v: integer;
  pixValue: SmallInt;
  k: real;
  RGBValues: array of byte;
begin
  if Self.Series.generalSeries.modality <> 'US' then
    begin
    WindowCenter:= newWindowCenter;
    WindowWidth:= newWindowWidth;
    bmp:= Picture.Bitmap;
    min:= WindowCenter - (WindowWidth div 2);
    max:= WindowCenter + (WindowWidth div 2);
    k:= (WindowCenter / WindowWidth) - 0.5;
    setLength(RGBValues, max - min + 1);
    for i:= 0 to max - min do
      RGBValues[i]:= trunc((((min + i)/WindowWidth) - k) * 256);
      //this formula means:
      //((pixValue - WC + (WW/2)) / WW) * 256;
    for i:= 0 to originalWidth - 1 do
      for j:= 0 to originalHeight - 1 do
      begin
      pixValue:= pixelValue[i,j];
      if pixValue <= min then v:= 0   else
      if pixValue >= max then v:= 255 else
                          v:= RGBValues[pixelValue[i,j] - min];
                          //it´s inside the window, let´s get its GrayScale value;
      bmp.Canvas.Pixels[j,i] := v or (v shl 8) or (v shl 16);
      end;
    end;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImage.processImageInfoElement(
  element: TCyclopsDataElement);
begin
  if converted then
    begin
    if Self.Series.generalSeries.modality = 'US' then
```

```
      getUSValuesFromDataElement(element, fpixelValue)
    else
      getHUValuesFromDataElement(element, fpixelValue);
    end;
  if tempIndex = -1 then
    tempIndex:= tempImageManager.addImage(self);
end;
//----------------------------------------------------------------------------
procedure TCyclopsImage.recoverPixelInformation;
var
  reader: TCyclopsDCMReader;
begin
  if not converted then
     begin
    reader:= TCyclopsDCMReader.createWithFile(FileName, false);
    converted:= true;
    processDataElement(reader.getImagePixelElement);
    reader.finalizeInstance;
    reader.Free;
    end;
end;
//----------------------------------------------------------------------------
function TCyclopsImage.isConverted: boolean;
begin
  result:= converted;
end;
//----------------------------------------------------------------------------
procedure TCyclopsImage.deleteDataElements;
begin
   DataElements.deleteAll;
   dataElements.Free;
   dataElements:= nil;
end;
//----------------------------------------------------------------------------
procedure TCyclopsImage.setImageNumber(number: integer);
begin
    fImageNumber := number;
     Self.Hint := 'image n. - '+ IntToStr(imageNumber);
    ShowHint := true;
end;
//----------------------------------------------------------------------------
//description : This met hod copies all current data object to tmpImg.
// Input:    none
// OutPut:  copies all current data to tmpImg

procedure TCyclopsImage.copy(tmpImg: tcyclopsImage);
var
              i              : integer;
    tam       : integer;
begin
    converted                                          := tmpImg.converted;
    pOriginalW idth                          := tmpImg.originalWidth;
    pOriginalHeight                          := tmpImg.OriginalHeight;
    pImageFileTipe                           := tmpImg.ImageFileTipe;
    pCurrentOrientation                      := tmpImg.CurrentOrientation;
    fSOPInstanceUID                          := tmpImg.SOPInstanceUID;
//    CopyMemory(tmpImg.fPixelValue, fPixelValue,262143);
              tam := Length(tmpImg.fpixelValue) - 1;
    SetLength(fpixelvalue, tam);
   for i := 0 to  tam do
       fpixelValue[i] := tmpImg.fPixelValue[i];

//    fpixelValue                                      := tmpImg.fpixelValue;
    fWindowCenter                            := tmpImg.WindowCenter;
    fWindowWidth                             := tmpImg.WindowWidth;
    fOriginalWindowCenter          := tmpImg.OriginalWindowCenter;
    fOriginalWindowWidth           := tmpImg.OriginalWindowWidth;
    ftempIndex                                       := tmpImg.tempIndex;
//    ftempImageManager                  := tmpImg.tempImageManager;
    fImageNumber                             := tmpImg.imageNumber;
```

```
   fSeries                                          := tmpImg.Series;
   fSliceThickness                          := tmpImg.SliceThickness;
   fSamplesPerPixel                 := tmpImg.SamplesPerPixel;
   fDataElements                            := tmpImg.DataElements;
   fFileName                                        := tmpImg.FileName;

//related problems: Comented properties are not yet copied. In future, if
//necessary, they must be enabled.
end;
//-------------------------------------------------------------------------
destructor TCyclopsImage.Destroy;
begin
  //FreeMemory(fpixelValue);
  inherited Destroy;
end;
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
//description:  change the current image appearance
procedure TCyclopsImage.changePalette(newWindowCenter,NewWindowWidth: integer);
type
        bytearr = array [0..0] of byte;
var
  i, j, min, max: integer;
  v                                             : integer;
  pixValue          : SmallInt;
  k                                             : real;
  RGBValues                         : array of byte;
  p                                             : ^bytearr;
  p2                                    : array[0..262143] of Byte;

begin

   if Self.Series.generalSeries.modality <> 'US' then //window isn´t
   begin
     WindowCenter:= newWindowCenter; //adjust window center and
     WindowWidth:= newWindowWidth;   //                                                  width


     min:= WindowCenter - (WindowWidth div 2);
     max:= WindowCenter + (WindowWidth div 2);
     k:= (WindowCenter / WindowWidth) - 0.5;
     setLength(RGBValues, max- min + 1);

     for i:= 0 to max - min do
       RGBValues[i]:= trunc((((min + i)/WindowWidth) - k) * 256);

     ////////////////////////////
     for j:= 0 to OriginalHeight - 1 do
       for i:= 0 to originalWidth - 1 do
       begin
          pixValue:= pixelValue[i,j];
          if pixValue <= min then v:= 0   else
          if pixValue >= max then v:= 255 else
                        v:= RGBValues[pixelValue[i,j] - min];
                        //it´s inside the window, let´s get its GrayScale value;
          p2[j +(i * originalWidth)] := v;

       end;
     ////////////////////////////
     SetBitmapBits(Picture.Bitmap.Handle,originalWidth*OriginalHeight,@p2);

          FreeMemory(@p);
     Refresh;
     ////////////////////////////
   end;


end;
//-------------------------------------------------------------------------
function TCyclopsImage.calcAllImageHUValues;
```

```
type
        bytearr = array [0..0] of byte;
var
  i, j, min, max : integer;
  v: integer;
  pixValue: SmallInt;
  k: real;
  RGBValues: array of byte;
  fbmp: TBitmap;
  //WC,WW : integer;
  hpal: HPALETTE;
  pe      : PALETTEENTRY;
  //p                                       : ^bytearr;
  p2                                : array[0..262143] of Byte;

begin
    if Self.Series.generalSeries.modality <> 'US' then //window isn´t
    begin
       fbmp:= TBitmap.Create;
       fbmp.PixelFormat := pf8bit;
        fbmp.Width  := originalWidth;
        fbmp.Height := OriginalHeight;

      min:= WindowCenter  - (WindowWidth div 2);
       max:= WindowCenter + (WindowWidth div 2);
       k:= (WindowCenter / WindowWidth) - 0.5;
        setLength(RGBValues, max- min + 1);
                        ////////////////////////////////////////
       pal := nil;
       GetMem(pal, sizeof(TLogPalette) + sizeof(TPaletteEntry) * (256));
       pal.palVersion := $300;
       pal.palNumEntries := 256;

       for i := 0 to 255 do
       begin
       GetPaletteEntries(fbmp.Palette,i,1,pe);
       pal.palPalEntry[i].peRed := i;
       pal.palPalEntry[i].peGreen := i;
       pal.palPalEntry[i].peBlue := i;
       end;
       hpal := CreatePalette(pal^);
       if hpal <> 0 then
         fbmp.Palette := hpal;
       ////////////////////////////////////////

       for i:= 0 to max - min do
         RGBValues[i]:= trunc((((min + i)/WindowWidth) - k) * 256);

       ////////////////////////////
       for j:= 0 to OriginalHeight - 1 do
         for i:= 0 to originalWidth - 1 do
         begin
            pixValue:= pixelValue[i,j];
            if pixValue <= min then v:= 0   else
           if pixValue >= max then v:= 255 else
                           v:= RGBValues[pixelValue[i,j] - min];
                           //it´s inside the window, let´s get its GrayScale value;
            p2[j +(i * originalWidth)] := v;

         end;
       ////////////////////////////
       SetBitmapBits(fbmp.Handle,originalWidth*OriginalHeight,@p2);
       ////////////////////////////
    end;
    result := fbmp;
end;
//---------------------------------------------------------------------------
end.
//-------------------------------------------------------------------------------------------------------
```

```
(*******************************************************************************
CLASS   : TCyclopsImageList
Author  :       Daniel Duarte Abdala (Caju)
Date    :       09/10/2001

Description     : Class TCyclopsImageList is a especialized class of TList that
responds to all ancestor messages and implements in addiction a new nested list
containing one instance (at the same index) of TPaintBoxEx to each image on the
list. This listof TPaintBoxEx are used to anage all graphics efects rendered
in original images. (To know more about how drawing or magnify one image see
uPaintBoxEx).


Atributes :
        paintBoxList    : this is a list (TList) that contains one reference to a
   instance of TPaintBoxEx to each item on the list.
   refOwner             : this is a reference to owner (graphical object) used to
   correct positionig of paintBoxEx´s on the screen.


Methods :
   name : CreateWith(AOwner : TWinControl);
   description: constructor methodused to initialize some things. At here we
   instanciate the paintBoxList.

   name : Destroy; override;
   description: destructor method used to free used memory by paintBoxList.

   name : Clear; override;
   Description: this method clear all the image list and the paintBoxList, but
   don´t delete the lists.

   name : prepare(index : Integer; Sb : TStatusPanel);
   description : this method is used to correct positioning the correspondent
   paintboxEx Above a image. The argument SB, is a reference to a scroll box
   (on ufImageViewer). This sb in where the images will be properly displayed.

   name : PBItems(Index: Integer): Pointer;
   description:

   name : flip(orientation : OrientationType);
   description :

        name : zoonSelected(factor : Integer);
   description :

   name : setTool(t : TDrawingTool);
   description :

   name : setColor(c : TColor);
   description :

   name : EraseSelected;
   description :

   name : currentImIndexUnderMouse(X,Y : Integer): Integer;
   description :

   name : setMagnifyLevel(level : TMagnify);
   description :

   name : setMagnifyWidth(wid : Integer);
   description :

   name : showMagnifyOnImage(b : boolean; i : TImage);
   description :

   name : ShowAllDrawings(b : Boolean);
   description :
```

Related Errors :

Need to by Done :

```
*******************************************************************************)
unit UCyclopsImageList;

interface

uses
        //delphi units
        Classes,Controls,Graphics,ComCtrls, Menus,ExtCtrls,StdCtrls,
  //cyclops units
  UPaintBoxEx,UCyclopsImage,VectorGraphicsNodeLibrary;

type
        TCyclopsImageList = class (TList)

private
        paintBoxList        :TList;
    refOwner                :TWinControl;

public
    constructor CreateWith(AOwner : TWinControl);
     destructor         Destroy; override;

    procedure           Clear; override;
    procedure           prepare(index : Integer; Sb : TStatusPanel; Sb2 : TStatusPanel;edtw,edtc : tedit);
    function PBItems(Index: Integer): Pointer;
    procedure           flip(orientation : OrientationType; all : boolean);
    procedure           zoonSelected(factor : Integer);
    procedure           setTool(t : TDrawingTool);
    procedure           setColor(c : TColor);
    procedure           EraseSelected;
    function currentImIndexUnderMouse(X,Y : Integer): Integer;

    procedure           setMagnifyLevel(level : Integer);
    procedure           setMagnifyWidth(wid : Integer);
    procedure           showMagnifyOnImage(b : boolean; i : TImage);

    procedure           ShowAllDrawings(b : Boolean);

    procedure           changePallete(wc,ww,index: integer);
    function ImageNumberToIndex(imgNumber : integer) : Integer;
end;

implementation

{ TCyclopsImageList }
//----------------------------------------------------------------------------
procedure TCyclopsImageList.changePallete(wc, ww, index: integer);
var
        im : TPaintBoxEx;
begin
        im := PBItems(index);
    im.image.changePalette(wc,ww);
    im.image.Repaint;
end;
//----------------------------------------------------------------------------
procedure TCyclopsImageList.Clear;
var
        imCount                         : integer;
    imRef                       :TCyclopsImage;
    paintBoxExRef     : TPaintBoxEx;
begin

        //clear the imageList and paintBoxList
```

```
            for imCount := 0 to Count- 1 do
    begin
                        imRef := Items[imCount];
       imRef.Destroy;
       paintBoxExRef := paintBoxList.Items[imCount];
                        paintBoxExRef.Destroy;
    end;
     inherited Clear;
    paintBoxList.Clear;
end;
//---------------------------------------------------------------------------
constructor TCyclopsImageList.CreateWith(AOwner : TWinControl);
begin
            inherited Create;
   //creates a list to hold all paintBoxEx related with each image
            paintBoxList := TList.Create;

    refOwner := AOwner;
end;
//---------------------------------------------------------------------------
function TCyclopsImageList.currentImIndexUnderMouse(X,
  Y: Integer): Integer;
var
            cont        : Integer;
    t,l,w,h   : Integer;
    refIm     : TCyclopsImage;
begin
    for cont := 0 to Count - 1 do
    begin
       refIm := Items[cont];
       t := refIm.Top;
       l := refIm.Left;
       w := refIm.Width;
       h := refIm.Height;
       if ((X >= l) and (Y >= t) and (X <= l+w) and (Y <= t+h)) then
       begin
                                    result := cont;
          exit;
       end;
    end;
    result :=-1;
end;
//---------------------------------------------------------------------------
destructor TCyclopsImageList.Destroy;
begin
            inherited Destroy;

end;
//---------------------------------------------------------------------------
procedure TCyclopsImageList.EraseSelected;
var
            im                      : TPaintBoxEx;
    cont      : Integer;
begin
            for cont := 0 to Count- 1 do
    begin
       im := PBItems(cont);
                        im.apagaSelecionados;
    end;

end;
//---------------------------------------------------------------------------
procedure TCyclopsImageList.flip(orientation: OrientationType; all : boolean);
var
            im                      : TPaintBoxEx;
    cont      : Integer;
begin
//top, right,botton,left
            for cont := 0 to Count- 1 do
    begin
```

```
                im := PBItems(cont);
      if all then
            im.rotate(orientation)
      else if (im.beBordered) then //exec the flip operation
            im.rotate(orientation);
   end;

end;
//-----------------------------------------------------------------------------
//description: this method take the image number and retreave the image index at
//imageList
//Input     : a integer meaning the image  DICOM number
//OutPut    : a integer meaning the image index int the imageList
function TCyclopsImageList.ImageNumberToIndex(imgNumber: integer): Integer;
var
            cycimg     : TCyclopsImage;
   cont     : integer;
begin
            for cont := 0 to Count- 1 do
            begin
                        cycimg := items[cont];
      if cycimg.imageNumber = imgNumber then
            result := cont;
   end;

end;
//-----------------------------------------------------------------------------
function TCyclopsImageList.PBItems(Index: Integer): Pointer;
begin
            result := paintBoxList.Items[index];
end;
//-----------------------------------------------------------------------------
procedure TCyclopsImageList.prepare(index: Integer;
                                                                                    sb : TStatusPanel;Sb2 :
TStatusPanel;
                        edtw,edtc : tedit);
var
            paintBoxEx1            : TpaintBoxEx;
begin

  //add a corespondent paintBox
  paintBoxEx1          := TPaintBoxEx.Create(refOwner);
  paintBoxEx1.Parent := refOwner;
  paintBoxEx1.image := self.Items[index];
  paintBoxEx1.tool:= dtSelectTool;
  paintBoxEx1.redesenhar := true;

  //saw to show on SB the current mouse position
  paintBoxEx1.ShowCurrentPosition := true;
  ////////////
  paintBoxEx1.DestinationPanel := sb;
  paintBoxEx1.ImageNumber := sb2;
  ////////////
  paintBoxEx1.refWW := edtw;
  paintBoxEx1.refWC := edtc;
  ////////////
  paintBoxList.Add(paintBoxEx1);
  //ajust paintBox to ref image
  paintBoxEx1.reposiciona;

end;
//-----------------------------------------------------------------------------
procedure TCyclopsImageList.setColor(c: TColor);
var
   cont     : integer;
   pbTemp: TPaintBoxEx;
begin


   for cont := 0 to paintBoxList.count - 1 do
```

```
    begin
            pbTemp := paintBoxList.Items[cont];
        pbTemp.Canvas.Pen.Color := c;
    end;

end;
//------------------------------------------------------------------------------
procedure TCyclopsImageList.setMagnifyLevel(level: Integer);
var
            cont        : Integer;
    ref     : TPaintBoxEx;
begin
            for cont := 0 to paintBoxList.Count- 1 do
    begin
            ref := paintBoxList.Items[cont];
        ref.MagnifyLevel := level;
    end;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImageList.setMagnifyWidth(wid: Integer);
var
            cont        : Integer;
    ref     : TPaintBoxEx;
begin
            for cont := 0 to paintBoxList.Count - 1 do
    begin
            ref := paintBoxList.Items[cont];
        ref.magnifyWidth := wid;
    end;

end;
//------------------------------------------------------------------------------
procedure TCyclopsImageList.setTool(t: TDrawingTool);
var
            pbTemp    : TPaintBoxEx;
    cont      : integer;
begin

    for cont := 0 to paintBoxList.count - 1 do
    begin
            pbTemp := paintBoxList.Items[cont];
        pbTemp.tool := t;
    end;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImageList.ShowAllDrawings(b: Boolean);
begin

end;
//------------------------------------------------------------------------------
procedure TCyclopsImageList.showMagnifyOnImage(b: boolean; i: TImage);
var
            pbTemp    : TPaintBoxEx;
    cont      : integer;
begin
    for cont := 0 to paintBoxList.count - 1 do
    begin
            pbTemp := paintBoxList.Items[cont];
        pbTemp.ImageDest := i;
    end;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImageList.zoonSelected(factor: Integer);
var
            pbTemp    : TPaintBoxEx;
    cont      : integer;
begin
    for cont := 0 to paintBoxList.count - 1 do
    begin
            pbTemp := paintBoxList.Items[cont];
```

```
      case factor of
      1      :
               begin
               pbTemp.Width := pbTemp.Width mod 8;
              pbTemp.Height := pbTemp.Height mod 8;
            end;
      2      :
               begin
               pbTemp.Width := pbTemp.Width mod 4;
             pbTemp.Height := pbTemp.Height mod 4;
            end;
      3      :
               begin
               pbTemp.Width := pbTemp.Width mod 2;
             pbTemp.Height := pbTemp.Height mod 2;
            end;
      4      :
               begin
//             pbTemp.Width := pbTemp.Width;
//              pbTemp.Height := pbTemp.Height;
            end;
      5      :
               begin
               pbTemp.Width := pbTemp.Width * 2;
             pbTemp.Height := pbTemp.Height * 2;
            end;
      6      :
               begin
               pbTemp.Width := pbTemp.Width * 4;
             pbTemp.Height := pbTemp.Height * 4;
            end;
      7      :
               begin
               pbTemp.Width := pbTemp.Width * 8;
             pbTemp.Height := pbTemp.Height * 8;
            end;
      end;
    end;

end;
//---------------------------------------------------------------------------
end.
//---------------------------------------------------------------------------------------------------------

unit uCyclopsImagesPrinter;

interface

uses PrintDrv, classes, ExtCtrls, SysUtils, Forms, graphics;

type TCyclopsImagesPrinter = class
protected
  printer: TMWPrintObject;
  FImagesPerLine: integer;
  FImagesXSpacing: single;
  FImagesYSpacing: single;
  FPrinting: boolean;
  FLeftMargin, FTopMargin, FRightMargin, FBottomMargin: single;
  FCyclopsLogo: TImage;
  FClinicLogo: TImage;
  FHeaderText: String;
  FHeaderHeight: single;
  pos        : array[1..9] of integer;

  function getPaperWidth: single;
  function getPaperHeight: single;
  function getWriteableWidth: single;
  function getFont: TFont;
  procedure setImagesPerLine(aValue: integer);
```

```
  procedure setImagesXSpacing(aValue: single);
  procedure setImagesYSpacing(aValue: single);
  procedure setLeftMargin(aValue: single);
  procedure setTopMargin(aValue: single);
  procedure setRightMargin(aValue: single);
  procedure setBottomMargin(aValue: single);
  procedure setFont(aValue: TFont);
public
  procedure beginDoc;
  procedure endDoc;

  constructor create(AOwner: TComponent);
  procedure printAllImagesFromHeight(images: TList; startHeight: single);
  procedure PrintImagesFromHeight(images: TList; startHeight: single; var unprintedImages: TList);
  procedure printImages(images: TList; var unprintedImages: TList);
  procedure writeHeader;
  procedure setMargins(left, top, right, bottom: single);
  procedure setHeader(x, y: single; w, h: single; boxed: boolean; shadeColor: TColor; text: TStrings);
  procedure newPage;
  procedure writeTextWrap(xl, xr, y: single; text: string);
  procedure writeInfoField(x, y: single; fieldName, fieldValue: string);
  procedure writeInfoFields(xl, xr,y: single; fieldNames, fieldValues: TStrings; var yPos: single);

  procedure writeImageReport(text: string; fieldNames, fieldValues: TStringList; images: Tlist);
  procedure previewImageReport(text: string; fieldNames, fieldValues: TStringList; images: TList);
  procedure imagespos(p :array of integer);

  property paperWidth:single read getPaperWidth;
  property paperHeight:single read getPaperHeight;
  property ImagesPerLine: integer read FImagesPerLine write setImagesPerLine;
  property Printing: boolean read FPrinting;
  property ImagesXSpacing: single read FImagesXSpacing write setImagesXSpacing;
  property ImagesYSpacing: single read FImagesYSpacing write setImagesYSpacing;
  property LeftMargin: single read FLeftMargin write setLeftMargin;
  property TopMargin: single read FTopMargin write setTopMargin;
  property RightMargin: single read FRightMargin write setRightMargin;
  property BottomMargin: single read FBottomMargin write setBottomMargin;
  property CyclopsLogo: TImage read FCyclopsLogo write FCyclopsLogo;
  property ClinicLogo: TImage read FClinicLogo write FClinicLogo;
  property HeaderText: String read FHeaderText write FHeaderText;
  property writeableWidth: single read getWriteableWidth;
  property HeaderHeight: single read FHeaderHeight write FHeaderHeight;
  property Font: TFont read getFont write setFont;
end;

implementation

{ TCyclopsImagesPrinter }
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.beginDoc;
begin
  printer.Start;
  Fprinting:= true;
  printer.SetMargins(topMargin, paperHeight - bottomMargin, leftMargin, paperWidth  - rightMargin);
end;
//------------------------------------------------------------------------------
constructor TCyclopsImagesPrinter.create(AOwner: TComponent);
var
  path: string;
begin
  path:= ExtractFilePath(application.exeName);
  printer:= TMWPrintObject.Create(AOwner);
  printer.Measurement:= mtMM;
  FImagesPerLine:= 1;
  FPrinting:= false;
  FImagesXSpacing:= 5.0;
  FImagesYSpacing:= 5.0;
  FHeaderHeight:= 0.0;
  CyclopsLogo:= TImage.Create(AOwner);
  CyclopsLogo.Picture.LoadFromFile(path + 'cyclops.bmp');
```

```
  ClinicLogo:= TImage.Create(AOwner);
  ClinicLogo.Picture.LoadFromFile(path + 'clinic.bmp');
  HeaderText:= 'Lista de Imagens';
  setMargins(20.0, 10.0, 20.0, 10.0);
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.endDoc;
begin
  printer.Quit;
  Fprinting:= false;
end;
//----------------------------------------------------------------------------
function TCyclopsImagesPrinter.getFont: TFont;
begin
  result:= printer.GetFont;
end;
//----------------------------------------------------------------------------
function TCyclopsImagesPrinter.getPaperHeight: single;
var
  h: single;
begin
  case printer.PaperSize of
    PAPER_A4: h:= 279.4;
  else
    h:= 279.4;
  end;
  result:= h;
end;
//----------------------------------------------------------------------------
function TCyclopsImagesPrinter.getPaperWidth: single;
var
  w: single;
begin
  case printer.PaperSize of
    PAPER_A4: w:= 215.9;
  else
    w:= 215.9;
  end;
  result:= w;
end;
//----------------------------------------------------------------------------
function TCyclopsImagesPrinter.getWriteableWidth: single;
begin
  result:= paperWidth - LeftMargin - RightMargin;
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.imagespos(p: array of integer);
var
        i : integer;
begin
        for i := 1 to 9 do
                pos[i] := p[i];
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.newPage;
begin
  printer.newPage;
  writeHeader;
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.previewImageReport(text: string;
  fieldNames, fieldValues: TStringList; images: TList);
//var
//  head: TStringList;
//  f: TFont;
//  y: single;
begin
  printer.Preview:= true;
  writeImageReport(text, fieldNames, fieldValues, images);
  printer.Preview:= false;
```

```
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.printAllImagesFromHeight(images: TList;
  startHeight: single);
var
  unprinted, imgs: TList;
begin
  PrintImagesFromHeight(images, startHeight, unprinted);
  while unprinted.Count > 0 do
    begin
    NewPage;
    imgs:= unprinted;
    printImages(imgs, unprinted);
    imgs.Free;
    end;
  unprinted.Free;
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.printImages(images: TList;
  var unprintedImages: TList);
begin
  //just start printing images after the header, giving some spacing
  PrintImagesFromHeight(images, topMargin + HeaderHeight + 10.0, unprintedImages);
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.PrintImagesFromHeight(images: TList;
  startHeight: single; var unprintedImages: TList);
var
  done: boolean;
  i,j, k: integer;
  x, y: single;
  imgsWidth, imgsHeight: single;
  ind: integer;
  img: TImage;
begin
  unprintedImages:= TList.Create;

  //this method must only be called when you already have started a printing job
  if not printing then
    exit;

  if images.Count = 0 then //no images to print, nothing to do...
    exit;

  done:= false;

  //get the images dimensions on the paper
  img:= images.first;
  imgsWidth:= (writeableWidth - ((imagesPerLine - 1) * imagesXSpacing)) / imagesPerLine;
  imgsHeight:= (imgsWidth / img.Picture.Bitmap.Width) * img.Picture.Bitmap.height;

  y:= startHeight; //this will be the Y position of the first row of Images

  //i means the printing row (starting at 0);
  for i:= 0 to images.Count div imagesPerLine do
    begin
    x:= leftMargin; //this will be the X position of the first column of Images
    for j:= 0 to imagesPerLine - 1 do
      begin
      ind:= (i * imagesPerLine) + j; //this is the index of the image to be printed now

      if (ind > (images.Count - 1)) then
      //if index is greater, all images have been printed and the job is done...
        begin
        done:= true;
        break;
        end // end if
      else
      //... if not, get the TImage and print it!
      img:= images.items[ind];
```

```
      printer.PrintGraphic(x, y, imgsWidth, imgsHeight, img);
      //prints the image "img" at (X,Y) , with the calculated dimensions;

      x:= x + imgsWidth + imagesXSpacing;
      //pushes X to the right of the last printed image,
      //giving the right spacing between them

      end; //end for j:=... (row printing)

    if done then //no more images to be printed, finished!
      break
    else
      begin
      y:= y + imgsHeight + imagesYSpacing;
      //pushes Y to the bottom of the last printed row,
      //giving the right spacing between them

      if ((y + imgsHeight) > (paperHeight- BottomMargin)) then
      //this means that the new row would be beyond the bottom margin
      //if so, put all the unprinted images in unprintedImages, and exit method
        begin
        for k:= ind  + 1 to images.Count- 1 do
          unprintedImages.Add(images.Items[k]);
        break;
        end; //end if (bottom margin check)
      end;// end else (if done)
    end; //enf for i:= ... (new rows);
end;
//---------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setBottomMargin(aValue: single);
begin
  if not printing then
    FBottomMargin:= aValue;
end;
//---------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setFont(aValue: TFont);
begin
  printer.SetFontInfo(aValue);
end;
//---------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setHeader(x, y, w, h: single;
  boxed: boolean; shadeColor: TColor; text: TStrings);
var
  i: integer;
  lh: single;
  py: single;
begin
  lh:= printer.GetLineHeightMM;
  printer.SetHeaderDim(x, y, x + w, y + h, boxed, 2, shadeColor);
  py:= y + ((h - (text.Count * lh)) / 2);
  for i:= 1 to text.Count do
    begin
    printer.SetHeaderInfo(i, py, text.Strings[i - 1], wtCenter, printer.getFont);
    py:= py + lh;
    end;
  FHeaderHeight:= h;
end;
//---------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setImagesPerLine(aValue: integer);
begin
  if not printing then
    FImagesPerLine:= aValue;
end;
//---------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setImagesxSpacing(aValue: single);
begin
  if not printing then
    FImagesXSpacing:= aValue;
end;
```

```
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setImagesYSpacing(aValue: single);
begin
  if not printing then
    FImagesYSpacing:= aValue;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setLeftMargin(aValue: single);
begin
  if not printing then
    FLeftMargin:= aValue;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setMargins(left, top, right,
  bottom: single);
begin
  if not printing then
    begin
    FLeftMargin:= left;
    FTopMargin:= top;
    FRightMargin:= right;
    FBottomMargin:= bottom;
    end;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setRightMargin(aValue: single);
begin
  if not printing then
    FRightMargin:= aValue;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.setTopMargin(aValue: single);
begin
  if not printing then
    FTopMargin:= aValue;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.writeHeader;
begin
  printer.PrintGraphic(leftMargin, topMargin, 0.14 * writeableWidth, headerHeight, CyclopsLogo);
  printer.PrintGraphic(leftMargin + (0.74 * writeableWidth), topMargin, 0.26 * writeableWidth, headerHeight, ClinicLogo);
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.writeImageReport(text: string; fieldNames,
  fieldValues: TStringList; images: TList);
var
  head: TStringList;
  f: TFont;
  y: single;
begin
  head:= TStringList.Create;
  head.Add('Lista de Imagens');
  f:= TFont.Create;
  f.Name:= 'Arial';
  beginDoc;
  Font:= f;
  setHeader(LeftMargin + (0.14 * writeableWidth), topMargin, (0.6 * writeableWidth), 15.0, true, $CFCFCF, head);
  writeHeader;

  writeTextWrap(leftMargin, paperWidth - rightMargin, topMargin + HeaderHeight + 5.0, text);
  y:= printer.GetYPosition + printer.GetLineHeightMM;
  y:= y + printer.GetLineHeightMM;
  writeInfoFields(leftMargin, paperWidth - rightMargin, y, fieldNames, fieldValues, y);
  printAllImagesFromHeight(images, y + 5.0);
  endDoc;
  head.Free;
end;
//------------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.writeInfoField(x, y: single; fieldName,
  fieldValue: string);
```

```
var
  xValue: single;
  oldStyle: TFontStyles;
begin
  oldStyle:= printer.GetFontStyle;
  printer.SetFontStyle(oldStyle + [fsBold]);
  printer.WriteLine(x, y, fieldName + ': ');
  xValue:= x + printer.GetTextWidthMM(fieldName + ': ');
  printer.SetFontStyle(oldStyle - [fsBold]);
  printer.WriteLine(xValue, y, fieldValue);
  printer.SetFontStyle(oldStyle);
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.writeInfoFields(xl, xr, y: single; fieldNames,
  fieldValues: TStrings; var yPos: single);
var
  vy: single;
  i: integer;
begin
  vy:= y;
  for i:= 0 to fieldNames.Count - 1 do
    begin
    vy:= vy + printer.GetLineHeightMM;
    writeInfoField(xl, vy, '   ' + fieldNames.Strings[i], fieldValues.Strings[i]);
    end;
  yPos:= vy + (2 * printer.GetLineHeightMM);
  printer.DrawBox(xl, y, xr, yPos, 3);
end;
//----------------------------------------------------------------------------
procedure TCyclopsImagesPrinter.writeTextWrap(xl, xr, y: single;
  text: string);
begin
  printer.WriteLineWrap(xl, xr, y, text, false, false);
end;
//----------------------------------------------------------------------------
end.

//-------------------------------------------------------------------------------------------------------------

unit uCyclopsLoadProgressForm;

interface

uses
  uCyclopsImage, Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ComCtrls, uCyclopsFileManager, ExtCtrls,
  Buttons, DateUtils,uCyclopsHandle;
  type EDicomFileError = class (Exception);
type

  TCyclopsLoadProgressForm = class(TForm)
    progress: TProgressBar;
    msg: TLabel;
    lFileName: TLabel;
    Panel1: TPanel;
    Image1: TImage;
    lElapsed: TLabel;
    lRemaining: TLabel;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    fastRead: boolean;
    totalFiles, filesRead: integer;
    fileNames: TStringList;
    FileManager: TCyclopsfileManager;
    completed: boolean;
    initialTime: TDateTime;
    lastFileTime: TDateTime;
  public
    { Public declarations }
```

```
    constructor createW ith(AOwner: TComponent; files: TStringList; fm: TCyclopsfileManager; vFastRead: boolean);
    procedure initalizeState(vtotalFiles:integer);
    procedure updateState(vfilesRead: integer; fileName: string);
    procedure estimateTime;
    procedure run;
  end;

var
  CyclopsLoadProgressForm: TCyclopsLoadProgressForm;

implementation

{$R *.dfm}

{ TCyclopsLoadProgressForm }
//-----------------------------------------------------------------------------
constructor TCyclopsLoadProgressForm.createWith(AOwner: TComponent;
     files: TStringList; fm: TCyclopsfileManager; vFastRead: boolean);
begin
  create(AOwner);
  fileNames:= files;
  FileManager:= fm;
  completed:= false;
  fastRead:= vFastRead;
end;
//-----------------------------------------------------------------------------
procedure TCyclopsLoadProgressForm.initalizeState(vtotalFiles: integer);
begin
  totalFiles:= vTotalFiles;
  msg.caption:= 'Arquivo 0 de ' + inttostr(totalFiles);
  progress.Max:= totalFiles;
end;
//-----------------------------------------------------------------------------
procedure TCyclopsLoadProgressForm.run;
var
  cont: integer;
  fileName: string;
begin
  //try
  initalizeState(FileNames.count);
  show;
  Application.ProcessMessages;
  initialTime:= now;
  lastFileTime:= now;
  try
    for cont:= 0 to FileNames.count - 1 do
      begin
      fileName := FileNames.Strings[cont];
      filesRead:= cont + 1;
      updateState(cont + 1, fileName);
//      FileManager.openDCMFileStudyInfo(fileName); //old fashion!!!!!
      FileManager.openDCMFile(fileName, FastRead);
      end;
    completed:= true;
  finally
    close;
  end;
end;
//-----------------------------------------------------------------------------
procedure TCyclopsLoadProgressForm.updateState(vfilesRead: integer; fileName: string);
var
  actualTime: TDateTime;
  timeElapsed, timePerFile, remainingTime: int64;
begin
  lFileName.Caption:= ExtractFileName(fileName);
  msg.Caption:= 'Arquivo ' + inttostr(vfilesRead) + ' de ' + inttostr(totalFiles);
  progress.position:= vFilesRead;
  if vfilesRead > 0 then
    begin
    actualTime:= now;
```

```
    timeElapsed:= MilliSecondsBetween(actualTime, initialTime);
    timePerFile:= MilliSecondsBetween(actualTime, lastFileTime);
    lastFileTime:= now;
//    timePerFile:= timeElapsed div vFilesRead;  //this didn´t worked very well...
    remainingTime:= (totalFiles - vFilesRead) * timePerFile; //this seems to work better
    lElapsed.caption:= 'Time elapsed: ' + inttostr(timeElapsed div 1000) + ' seconds';
    lRemaining.caption:= 'Time remaining: ' + inttostr(remainingTime div 1000) + ' seconds';
    end;
  refresh;
end;
//---------------------------------------------------------------------------
procedure TCyclopsLoadProgressForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if completed then
    action:= caFree
  else
    action:= caNone
end;
//---------------------------------------------------------------------------
procedure TCyclopsLoadProgressForm.estimateTime;
begin

end;
//---------------------------------------------------------------------------
end.

//-----------------------------------------------------------------------------------------------------------

unit uCyclopsPatient;

interface

uses uCyclopsPatientModule, uCyclopsDataElement, classes, uCyclopsgeneralFunctions;

type TCyclopsPatient = class
  protected
  patientModule: TCyclopsPatientModule;
  fstudies: TList;
  FFiles: TStringList;
  public
  class procedure initialize;
  class function needsTag(groupNumber, elementNumber: word): boolean;

  constructor create;

  procedure processDataElement(element: TCyclopsDataElement);
  procedure addStudy(study: TObject);

  property data: TCyclopsPatientModule read patientModule write patientModule;
  property studies : TList read fstudies write fstudies;
  property Files : TStringList read FFiles write FFiles;
end;

var
  PatientTags: array[0..3, 0..1] of word;
implementation

{ TCyclopsPatient }
//---------------------------------------------------------------------------
procedure TCyclopsPatient.addStudy(study: TObject);
begin
  studies.add(study);
end;
//---------------------------------------------------------------------------
constructor TCyclopsPatient.create;
begin
  studies:= TList.create;
  patientModule := TCyclopsPatientModule.Create;
  Files:= TStringList.Create;
```

```
end;
//-------------------------------------------------------------------------------
class procedure TCyclopsPatient.initialize;
begin
//      Group Number            Element Number
//------------------------ -------------------------
  PatientTags[0,0]:= $0010;  PatientTags[0,1]:= $0010;
  PatientTags[1,0]:= $0010;  PatientTags[1,1]:= $0020;
  PatientTags[2,0]:= $0010;  PatientTags[2,1]:= $1005;
  PatientTags[3,0]:= $0010;  PatientTags[3,1]:= $0040;
end;
//-------------------------------------------------------------------------------
class function TCyclopsPatient.needsTag(groupNumber,
  elementNumber: word): boolean;
var
  i: integer;
  match: boolean;
begin
  match:= false;
  for i:= 0 to high(PatientTags) do
    begin
   if (groupNumber   = PatientTags[i, 0]) and
     (elementNumber = PatientTags[i, 1]) then
     begin
     match:= true;
     break;
     end;
   end;
  result:= match;
end;
//-------------------------------------------------------------------------------
procedure TCyclopsPatient.processDataElement(element: TCyclopsDataElement);
begin
  //check if the DE has the right group number
  case element.ElementNumber of
   $0010: data.patientsName:= TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
   $0020: data.patientID:= TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
   $0030: data.patientsBirthDate:= TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
   $0040: data.patientsSex:= TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
  end;
end;
//-------------------------------------------------------------------------------
end.

//-----------------------------------------------------------------------------------------------------

(*          CLASS TCyclopsPatientModule
            Author:             Daniel Duarte (caju)
   Date:            29/08/2001
   Description: Definition of patient's atributes

other: All defined atribute has his tag as comment

*******************************************************************************)
unit UCyclopsPatientModule;

interface
uses
            //Delphi Units
            Controls,
   //Cyclops Units
   UCyclopsIEDefinitions;
            type TCyclopsPatientModule = class

private
   fpatientsName            : string;
   fpatientID                        : string;
   fpatientsBirthDate : string;
   fpatientsSex             : string;
protected
```

```
public

published //accessing methods

  property  patientsName: string read fpatientsName write fpatientsName;              //(0010,0010)
  property  patientID  : string read fpatientID write fpatientID;                     //(0010,0020)
  property  patientsBirthDate : string read fpatientsBirthDate write fpatientsBirthDate; //(0010,1005)
  property  patientsSex : string read fpatientsSex write fpatientsSex;                //(0010,0040)
end;

implementation

end.

//-----------------------------------------------------------------------------------------------------

unit UCyclopsPatientStudyModule;

interface
        type TCyclopsPatientStudyModule = class

private
   _patientsAge        : string;
   _patientsSize       : string;
   _patientsWeight     : string;
protected

public

published
        property patientsAge : string read _patientsAge write _patientsAge;
   property patientsSize : string read _patientsSize write _patientsSize;
   property patientsWeight : string read _patientsWeight write _patientsWeight;
end;
implementation

end.
//-----------------------------------------------------------------------------------------------------

unit uCyclopsSeries;

//---------------------------------------------------------------
//  This is a  Smalltalk class converted with
//  the ToDelphiConversor, created by Charles I. Wust,
//  member of The Cyclops Project.
//---------------------------------------------------------------

interface

uses classes, uCyclopsGeneralSeriesModule, uCyclopsStudy, uCyclopsDataElement,
uCyclopsGeneralFunctions;

type TCyclopsSeries = class

protected
  Fequipment : string;
//  FframeOfReference : string;
  FgeneralSeries : TCyclopsGeneralSeriesModule;
  FicoBitsPerPixel : string;
  FicoDepth : string;
  FicoHeight : string;
  FicoWidth : string;
  Fimages : TList;
  FimgBitsPerPixel : string;
  FimgDepth : string;
  FimgHeight : string;
  FimgWidth : string;
//  Fpalette : string;
  Fstudy : TCyclopsStudy;
```

```
  FFiles: TStringList;
public
  constructor create;
  procedure processDataElement(element: TCyclopsDataElement);
  function getImages: TList;
  procedure convertAllImages;
  procedure convertImageAtIndex(index : integer);
  property equipment : string read Fequipment write Fequipment;
//  property frameOfReference : string read FframeOfReference write FframeOfReference;
  property generalSeries : TCyclopsGeneralSeriesModule read FgeneralSeries write FgeneralSeries;
  property icoBitsPerPixel : string read FicoBitsPerPixel write FicoBitsPerPixel;
  property icoDepth : string read FicoDepth write FicoDepth;
  property icoHeight : string read FicoHeight write FicoHeight;
  property icoWidth : string read FicoWidth write FicoWidth;
  property images : TList read getImages write Fimages;
  property imgBitsPerPixel : string read FimgBitsPerPixel write FimgBitsPerPixel;
  property imgDepth : string read FimgDepth write FimgDepth;
  property imgHeight : string read FimgHeight write FimgHeight;
  property imgWidth : string read FimgWidth write FimgWidth;
//  property palette : string read Fpalette write Fpalette;
  property study : TCyclopsStudy read Fstudy write Fstudy;
  property Files: TStringList read FFiles write FFiles;
end;

function compareImages(img1: Pointer; img2: Pointer): integer;

implementation

{ TCyclopsSeries }
uses uCyclopsImage;

function compareImages(img1: Pointer; img2: Pointer): integer;
var
  image1, image2: TCyclopsImage;
  ret: integer;
begin
  image1:= img1;
  image2:= img2;
  if image1.imageNumber > image2.imageNumber then
    ret:= 1
  else
    if image1.imageNumber < image2.imageNumber then
      ret:= -1
    else
      ret:= 0;
  result:= ret;
end;

procedure TCyclopsSeries.convertAllImages;
var
  i: integer;
  image: TCyclopsImage;
begin
  for i:= 0 to images.Count - 1 do
    begin
    image:= fImages.items[i];
    image.recoverPixelInformation;
    end;
end;
//---------------------------------------------------------------------------
procedure TCyclopsSeries.convertImageAtIndex(index: integer);
var
        image       : TCyclopsImage;
begin
        image := Fimages.Items[index];
    image.recoverPixelInformation;
end;
//---------------------------------------------------------------------------
constructor TCyclopsSeries.create;
begin
```

```
  images:= TList.Create;
  generalSeries:= TCyclopsGeneralSeriesModule.Create;
  Files:= TStringList.create;
end;

function TCyclopsSeries.getImages: TList;
begin
  fImages.Sort(@compareImages);
  result:= fImages;
end;

procedure TCyclopsSeries.processDataElement(element: TCyclopsDataElement);
begin
  case element.GroupNumber of
    $0008:
      case element.ElementNumber of
        $0021: generalSeries.seriesDate := TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $0031: generalSeries.seriesTime := TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $0060: generalSeries.modality:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $103E: generalSeries.seriesDescription:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $1050: generalSeries.performingPhysiciansName:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $1070: generalSeries.operatorsName:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
      end;
    $0018:
      case element.ElementNumber of
        $0015: generalSeries.bodyPartExamined:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $5100: generalSeries.patientPosition := TCyclopsGeneralFunctions.byteArrayAsString(element.value);
      end;
    $0020:
      case element.ElementNumber of
        $000E: generalSeries.seriesInstanceUID:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
        $0011: generalSeries.seriesNumber:= TCyclopsGeneralFunctions.byteArrayAsString(element.value);
      end;
  end;
end;

end.

//-----------------------------------------------------------------------------------------------------

unit uCyclopsStudy;

//---------------------------------------------------------------
//  This is a Smalltalk class converted with
//  the ToDelphiConversor, created by Charles I. Wust,
//  member of The Cyclops Project.
//---------------------------------------------------------------

interface

uses classes, uCyclopsGeneralStudyModule, uCyclopsPatient, uCyclopsPatientStudyModule,
              uCyclopsDataElement,uCyclopsgeneralFunctions;

type T CyclopsStudy = class

protected
  FgeneralStudy : TCyclopsGeneralStudyModule;
  Fpatient : TCyclopsPatient;
  FpatientStudy : TCyclopsPatientStudyModule;
  Fseries : TList;
  FFiles: TStringList;
public
              constructor create();
  procedure processDataElement(element: TCyclopsDataElement);

  property generalStudy : TCyclopsGeneralStudyModule read FgeneralStudy write FgeneralStudy;
  property patient : TCyclopsPatient read Fpatient write Fpatient;
  property patientStudy : TCyclopsPatientStudyModule read FpatientStudy write FpatientStudy;
  property series : TList read Fseries write Fseries;
  property Files : TStringList read FFiles write FFiles;
```

```
end;

implementation

{TCyclopsStudy}

{ TCyclopsStudy }
//---------------------------------------------------------------------------
constructor TCyclopsStudy.create;
begin
  series := TList.create;
  patientStudy := TCyclopsPatientStudyModule.Create;
  generalStudy := TCyclopsGeneralStudyModule.Create;
  Files:= TStringList.Create;
end;
//---------------------------------------------------------------------------
procedure TCyclopsStudy.processDataElement(element: TCyclopsDataElement);
begin
  //check if the DE has the right group number
  case element.GroupNumber of
    $0008:
      case element.ElementNumber of
        $0020: FgeneralStudy.studyDate := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
        $0030: FgeneralStudy.studyTime := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
        $0090: FgeneralStudy.referringPhysiciansName := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
        $0050: FgeneralStudy.accessionNumber := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
        $1010: FpatientStudy.patientsAge := TCyclopsgeneralFunctions.byteArrayAsStrin g(element.Value);
        $1030: FgeneralStudy.studyDescription := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
        $1060: FgeneralStudy.nameOfPhysicianReadingStudy:= TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
      end;
    $0010:
      case element.ElementNumber of
        $1030: FpatientStudy.patientsWeight:= TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
      end;
    $0020:
      case element.ElementNumber of
        $000D: FgeneralStudy.studyInstanceUID := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
        $0010: FgeneralStudy.studyID := TCyclopsgeneralFunctions.byteArrayAsString(element.Value);
      end;
  end;
end;
//---------------------------------------------------------------------------

end.
//---------------------------------------------------------------------------------------------------------
unit UCyclopsTagsDictionary;

interface

uses classes;

type TTagsDictionary = class
  protected
  tagNumbers: TStringList;
  tagNames: TStringList;
  tagVRs: TStringList;
  public
  constructor create;
  procedure addTag(vTagNumber, vTagName, vTagVR: string);
  function returnName(vTagNumber: string): string;
end;

implementation

{ TTagsDictionary }

procedure TTagsDictionary.addTag(vTagNumber, vTagName, vTagVR: string);
begin
  tagNumbers.Add(vTagNumber);
  tagNames.Add(vTagName);
```

```
  tagVRs.Add(vTagVR);
end;

constructor TTagsDictionary.create;
begin
  tagNumbers:= TStringList.Create;
  tagNames:= TStringList.Create;
  tagVRs:= TStringList.Create;
  addTag('(0008,0001)', 'Length to End', 'XX');
  addTag('(0008,0005)', 'Specific Character Set', 'CS');
  addTag('(0008,0008)', 'Image Type', 'CS');
  addTag('(0008,0010)', 'Recognition Code', 'XX');
  addTag('(0008,0012)', 'Instance Creation Date', 'DA');
  addTag('(0008,0013)', 'Instance Creation Time', 'TM');
  addTag('(0008,0014)', 'Instance Creator UID', 'UI');
  addTag('(0008,0016)', 'SOP Class UID', 'UI');
  addTag('(0008,0018)', 'SOP Instance UID', 'UI');
  addTag('(0008,0020)', 'Study  Date', 'DA');
  addTag('(0008,0021)', 'Series Date', 'DA');
  addTag('(0008,0022)', 'Acquisition Date', 'DA');
  addTag('(0008,0023)', 'Content Date', 'DA');
  addTag('(0008,0024)', 'Overlay Date', 'DA');
  addTag('(0008,0025)', 'Curve Date', 'DA');
  addTag('(0008,002A)', 'Acquisition Datetime', 'DT');
  addTag('(0008,0030)', 'Study Time', 'TM');
  addTag('(0008,0031)', 'Series Time', 'TM');
  addTag('(0008,0032)', 'Acquisition Time', 'TM');
  addTag('(0008,0033)', 'Content Time', 'TM');
  addTag('(0008,0034)', 'Overlay Time', 'TM');
  addTag('(0008,0035)', 'Curve Time', 'TM');
  addTag('(0008,0040)', 'Data Set Type', 'XX');
  addTag('(0008,0041)', 'Data Set Subtype', 'XX');
  addTag('(0008,0042)', 'Nuclear Medicine Series Type', 'CS');
  addTag('(0008,0050)', 'Accession Number', 'SH');
  addTag('(0008,0052)', 'Query/Retrieve Level', 'CS');

end;

function TTagsDictionary.returnName(vTagNumber: string): string;
var
  ind: integer;
begin
  ind:= tagNumbers.IndexOf(vTagNumber);
  if ind >= 0 then
    result:= tagNames.Strings[ind]
  else
    result:= '';
end;

end.
//----------------------------------------------------------------------------------------------------
unit uCyclopsTempImagesManager;

interface

uses Forms, classes, graphics, sysUtils;

type TCyclopsTempImagesManager = class

protected
  fTempDir: string;
  fLastIndex: integer;
  CyclopsImages: array [0..32000] of Pointer;
public
  constructor createOn(vTempDir: string);
  function addImage(aCyclopsImage: TObject): integer;
  function returnBitmap(aIndex: integer): TBitmap;

  property tempDir: string read fTempDir write fTempDir;
  property LastIndex: integer read fLastIndex write fLastIndex;
```

```
end;

implementation

uses uCyclopsImage;
{ TCyclopsTempImagesManager }

function TCyclopsTempImagesManager.addImage(
  aCyclopsImage: TObject): integer;
var
  tempIndex: integer;
  fbmp: TBitmap;
begin
  if aCyclopsImage.ClassNameIs('TCyclopsImage') then
     begin
     cyclopsImages[lastIndex]:= aCyclopsImage;
     lastIndex:= lastIndex + 1;
     tempIndex := lastIndex;
     fbmp:= (aCyclopsImage as TCyclopsImage).getOriginalBitmap;
     if fbmp <> nil then
       fbmp.SaveToFile(tempDir + '\tmp' + inttostr(tempIndex) + '.bmp');
     end
  else
     begin
     tempIndex:= -1;
     //throw exception: not a Cyclops Image
     end;
  result:= tempIndex;
end;

constructor TCyclopsTempImagesManager.createOn(vTempDir: string);
begin
  tempDir:= vTempDir;
  lastIndex:= 0;
end;

function TCyclopsTempImagesManager.returnBitmap(aIndex: integer): TBitmap;
var
  bmp: TBitmap;
//  fbmp: TBitmap;
  fileName: string;
//  cycImage: TCyclopsImage;
begin
  fileName:= tempDir + '\tmp' + inttostr(aIndex) + '.bmp';
  if not fileExists(fileName) then
     fileName:= ExtractFilePath(Application.ExeName) + 'notconverted.bmp';
  bmp:= TBitmap.Create;
  bmp.LoadFromFile(fileName);
  result:= bmp;
end;

end.

//----------------------------------------------------------------------------------------------------------

unit UDICOMEditorMain;

interface

uses
  Windows, Messages, SysUtils, Variants,Classes, Graphics, Controls, Forms,
  Dialogs, Grids, ExtCtrls, Menus, StdCtrls, uCyclopsGeneralFunctions,
  UCyclopsFilemanager, UCyclopsPatient, UCyclopsStudy, uCyclopsSeries,
  uCyclopsLoadProgressForm, uCyclopsImage, uImageViewer, ufOptions, ufQuerytool,
  ufServerStarter, ufAbout, uCyclopsImagesPrinter, uFPrintImages, uThumbnails,
  uMiniView, ufDirectoryFilter, uMailler, AppEvnts, uCyclopsHandle;

const
  annonymous = false;
```

```
type
  TfCyclopsDICOMEditorMain = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    MainMenu1: TMainMenu;
    Shape1: TShape;
    Shape2: TShape;
    Shape3: TShape;
    sgPatient: TStringGrid;
    sgStudies: TStringGrid;
    sgSeries: TStringGrid;
    sbThumbImages: TScrollBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Panel6: TPanel;
    Fille1: TMenuItem;
    Help1: TMenuItem;
    N2: TMenuItem;
    Close1: TMenuItem;
    Open1: TMenuItem;
    OpenSerie1: TMenuItem;
    popSeries: TPopupMenu;
    popStudies: TPopupMenu;
    popPatient: TPopupMenu;
    popThumbies: TPopupMenu;
    OpenSingle: TOpenDialog;
    OpenMulti: TOpenDialog;
    Open2: TMenuItem;
    N3: TMenuItem;
    ExportSerieas1: TMenuItem;
    OpenPatientStudies1: TMenuItem;
    N4: TMenuItem;
    ShowPatientInformation1: TMenuItem;
    OpenStudySeries1: TMenuItem;
    N5: TMenuItem;
    ShowStudyInformation1: TMenuItem;
    Server1: TMenuItem;
    ShowServerStatus1: TMenuItem;
    About1: TMenuItem;
    Panel7: TPanel;
    ServerOptions1: TMenuItem;
    N6: TMenuItem;
    OpenQueryTool1: TMenuItem;
    Image1: TImage;
    PrintImage1: TMenuItem;
    printDlg: TPrintDialog;
    PrintSeries1: TMenuItem;
    openwithminiviewer1: TMenuItem;
    pppImage: TPopupMenu;
    openWithMiniViewer2: TMenuItem;
    RecoverImageInformation1: TMenuItem;
    openmailler1: TMenuItem;
    ApplicationEvents1: TApplicationEvents;
    procedure FormCreate(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure OpenSerie1Click(Sender: TObject);
    procedure Close1Click(Sender: TObject);
    procedure sgPatientSelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
    procedure sgStudiesSelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
    procedure sgSeriesSelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
    procedure OpenPatientStudies1Click(Sender: TObject);
    procedure OpenStudySeries1Click(Sender: TObject);
    procedure Open2Click(Sender: TObject);
```

```
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormResize(Sender: TObject);
    procedure ShowServerStatus1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure ServerOptions1Click(Sender: TObject);
    procedure OpenQueryTool1Click(Sender: TObject);
    procedure sgSeriesContextPopup(Sender: TObject; MousePos: TPoint;
      var Handled: Boolean);
    procedure PrintImage1Click(Sender: TObject);
    procedure PrintSeries1Click(Sender: TObject);
    procedure openwithminiviewer1Click(Sender: TObject);
    procedure openWithMiniViewer2Click(Sender: TObject);
    procedure sgImagesSelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
    procedure RecoverImageInformation1Click(Sender: TObject);
    procedure openmailler1Click(Sender: TObject);
    procedure sgSeriesDblClick(Sender: TObject);
    procedure ApplicationEvents1Exception(Sender: TObject; E: Exception);
  private
    fm  : TCyclopsFileManager;
    selectedPatient: TCyclopsPatient;
    selectedStudy: TCyclopsStudy;
    selectedSerie: TCyclopsSeries;
    thumbs: TList;
//    searchPaths: TStringList;
    cyclopsPrinter: TCyclopsImagesPrinter;
//    selectedImages: TList;
    thumbsManager: TThmubnailsManager;

//    teste                 : integer;


    procedure createPatientCols;
    procedure createStudiesCols;
    procedure createSeriesCols;
    procedure showAllPatients;
    procedure showPatientStudy(aPatient: TCyclopsPatient);
    procedure showStudySeries(aStudy: TCyclopsStudy);
    procedure showSerieImages(aSerie: TCyclopsSeries);
    procedure showSerieImagesStepByStep(aCyclopsImage: TCyclopsImage);
//    procedure showImagesInfo(aSerie: TCyclopsSeries);
  public
    function openDirectory(path: string): TStringList;
    procedure clearTempDir;
    procedure recoverImageInfo;
  end;

var
  fCyclopsDICOMEditorMain: TfCyclopsDICOMEditorMain;

implementation

uses UCyclopsPatientModule, uCyclopsGeneralStudyModule;

{$R *.dfm}

{ TForm1 }
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.createPatientCols;
begin
    sgPatient.Cells[0,0] := 'Patient"s Name';
    sgPatient.Cells[1,0] := 'Patient ID';
    sgPatient.Cells[2,0] := 'Patient"s Birth Date';
    sgPatient.Cells[3,0] := 'Patient"s Sex';
end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.createSeriesCols;
begin
    sgSeries.Cells[0,0] := 'Modality';
    sgSeries.Cells[1,0] := 'Series Description';
```

```
    sgSeries.Cells[2,0] := 'Series Instance UID';
    sgSeries.Cells[3,0] := 'Series Number';
    sgSeries.Cells[4,0] := 'Series Date';
    sgSeries.Cells[5,0] := 'Series Time';
    sgSeries.Cells[6,0] := 'Performing Physician"s Name';
              sgSeries.Cells[7,0] := 'Operators" Name';
    sgSeries.Cells[8,0] := 'Body Part Examined';
    sgSeries.Cells[9,0] := 'Patient Position';

end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.createStudiesCols;
begin

    sgStudies.Cells[0,0] := 'Study Description';
    sgStudies.Cells[1,0] := 'Study Date';
    sgStudies.Cells[2,0] := 'Study Time';
    sgStudies.Cells[3,0] := 'Accession Number';
    sgStudies.Cells[4,0] := 'Referring Physician"s Name';
    sgStudies.Cells[5,0] := 'Study Instance UID';
    sgStudies.Cells[6,0] := 'Name of Physician(s) Reading Study';
    sgStudies.Cells[7,0] := 'Patient"s Age';
    sgStudies.Cells[8,0] := 'Patient"s Weight';

end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.FormCreate(Sender: TObject);
begin
    fm := TCyclopsFileManager.create;
    cyclopsPrinter:= TCyclopsImagesPrinter.create(self);
    thumbsManager:= TThmubnailsManager.createOn(sbThumbImages, popThumbies);
    createPatientCols;
    createStudiesCols;
    createSeriesCols;
    selectedPatient:= nil;
    selectedStudy:= nil;
    thumbs:= TList.Create;
    sbThumbImages.HorzScrollBar.Visible:= true;
    clearTempDir;

end;
//------------------------------------------------------------------------------

procedure TfCyclopsDICOMEditorMain.Open1Click(Sender: TObject);
var
  i: integer;
  Allfiles, files: TStringList;
  frmLoadProgress: TCyclopsLoadProgressForm;
  searchPaths: TStringList;
  dirFilter: TfDirectoryFilter;
begin
              try try
        searchPaths:= TStringList.Create;
        dirFilter:= TfDirectoryFilter.createWith(self, searchPaths);
        dirFilter.ShowModal;
        Allfiles:= TStringList.Create;
        if searchPaths.Count = 0 then exit;
        for i:= 0 to searchPaths.Count- 1 do
                  begin
                              files:= openDirectory(searchPaths.Strings[i]);
                              allFiles.AddStrings(files);
                              files.Free;
                            end;
        frmLoadProgress:= TCyclopsLoadProgressForm.createWith(self, allfiles, fm, true);
                              frmLoadProgress.run;
                              showAllPatients;
    finally
        showAllPatients;
        frmLoadProgress.Hide;
        frmLoadProgress.Close;
```

```
    end;
    except
            on E : TCyclopsHandle do
          E.logError;
    end;
end;
//-------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain .OpenSerie1Click(Sender: TObject);
var
    resp     : Boolean;
    open                    : TOpenDialog;
    frmLoadProgress: TCyclopsLoadProgressForm;
    Files: TStringList;
begin
            try try
        open := TOpenDialog.Create(self);
        open.Options := [ofAllowMultiSelect, ofReadOnly];
        open.Filter := 'DICOM files (*.dcm)|*.DCM';
        resp:= (open.Execute);
        if resp then
        begin
          files:= TStringList.Create;
          files.AddStrings(open.Files);
          frmLoadProgress:= TCyclopsLoadProgressForm.createWith(self, files, fm, false);
          frmLoadProgress.run;
          showAllPatients;
        end else
        begin
        //error handing
        end;
    finally
            showAllPatients;
        frmLoadProgress.Hide;
        frmLoadProgress.Close;
    end;
    except
            on E:TCyclopsHandle do
                    E.logError;
    end;
end;
//-------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.Close1Click(Sender: TObject);
begin
      Application.Terminate;
end;
//-------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.showPatientStudy(aPatient: TCyclopsPatient);
var
  cont: integer;
  study: TCyclopsStudy;
begin
  if aPatient <> nil then
    begin
    sgStudies.RowCount:= aPatient.studies.Count + 1;
    for cont := 0 to aPatient.studies.Count- 1 do
      begin
      study := aPatient.studies.Items[cont];
      sgStudies.Cells[0,cont + 1] := study.generalStudy.studyDescription;
      sgStudies.Cells[1,cont + 1] := study.generalStudy.studyDate;
      sgStudies.Cells[2,cont + 1] := study.generalStudy.studyTime;
      sgStudies.Cells[3,cont + 1] := study.generalStudy.accessionNumber;
      sgStudies.Cells[4,cont + 1] := study.generalStudy.referringPhysiciansName;
      sgStudies.Cells[5,cont + 1] := study.generalStudy.studyInstanceUID;
      sgStudies.Cells[6,cont + 1] := study.generalStudy.nameOfPhysicianReadingStudy;
      sgStudies.Cells[7,cont + 1] := study.patientStudy.patientsAge;
      sgStudies.Cells[8,cont + 1] := study.patientStudy.patientsWeight;
      end;
    end;
  sgSeries.rowCount:= 2;
  sgSeries.Rows[1].Clear;
```

```
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.sgPatientSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
begin
  if (fm.PatientsList.Count > 0) and
     (selectedPatient <> fm.PatientsList.items[aRow - 1]) then
    begin
    selectedPatient:= fm.PatientsList.items[aRow - 1];
    showPatientStudy(selectedPatient);
    selectedStudy:= selectedPatient.studies.first;
    sgStudies.Row:= 1;
    selectedSerie:= nil;
    sgSeries.RowCount:= 2;
    sgSeries.Rows[1].Clear;
    thumbsManager.clearThumbs;
    end;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.showStudySeries(aStudy: TCyclopsStudy);
var
  cont: integer;
  series: TCyclopsSeries;
begin
  if aStudy <> nil then
    begin
    sgSeries.RowCount:= aStudy.series.Count + 1;
    for cont := 0 to aStudy.series.Count - 1 do
      begin
      series:= aStudy.series.items[cont];
      sgSeries.Cells[0,cont + 1] := series.generalSeries.modality;
      sgSeries.Cells[1,cont + 1] := series.generalSeries.seriesDescription;
      sgSeries.Cells[2,cont + 1] := series.generalSeries.seriesInstanceUID;
      sgSeries.Cells[3,cont + 1] := series.generalSeries.seriesNumber;
      sgSeries.Cells[4,cont + 1] := series.generalSeries.seriesDate;
      sgSeries.Cells[5,cont + 1] := series.generalSeries.seriesTime;
      sgSeries.Cells[6,cont + 1] := series.generalSeries.performingPhysiciansName;
      sgSeries.Cells[7,cont + 1] := series.generalSeries.operatorsName;
      sgSeries.Cells[8,cont + 1] := series.generalSeries.bodyPartExamined;
      sgSeries.Cells[9,cont + 1] := series.generalSeries.patientPosition;
      end;
    sgSeries.Row:= 1;
//    selectedSerie:= selectedSerie:= selectedStudy.series.First;
    end;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.sgStudiesSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
begin
  if (selectedPatient <> nil) and
     (selectedPatient.studies.Count > 0) then
    begin
    selectedStudy:= selectedPatient.studies.items[aRow - 1];
    showStudySeries(selectedStudy);
    selectedSerie:= selectedStudy.series.first;
    sgSeries.Row:= 1;
    thumbsManager.clearThumbs;
    end;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.showAllPatients;
var
  cont: integer;
  patient: TCyclopsPatient;
begin
  sgPatient.RowCount:= fm.PatientsList.Count + 1;
  for cont := 0 to fm.PatientsList.Count - 1 do
  begin
    patient := fm.PatientsList.Items[cont];
    if annonymous then
```

```
      sgPatient.Cells[0,cont + 1] := 'Annonymous'
    else
      sgPatient.Cells[0,cont + 1] := patient.data.patientsName;
    sgPatient.Cells[1,cont + 1] := patient.data.patientID;
    sgPatient.Cells[2,cont + 1] := patient.data.patientsBirthDate;
    sgPatient.Cells[3,cont + 1] := patient.data.patientsSex;
  end;
end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.sgSeriesSelectCell(Sender: TObject;
  ACol, ARow: Integer; var CanSelect: Boolean);
begin
  if (selectedStudy <> nil) and
     (selectedStudy.series.Count > 0) then
    begin
    selectedSerie:= selectedStudy.series.items[aRow - 1];
    showSerieImages(selectedSerie);
    end;
end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.showSerieImages(aSerie: TCyclopsSeries);
var
//    refIm                 : TImage;
    cont                  : Integer;
//    refBmp                : Graphics.TBitmap;
//    lastTop               : Integer;
    refCycImage           : TCyclopsImage;
//    size: integer;
begin
  sbThumbImages.DisableAutoRange;
  thumbsManager.clearThumbs;
  for cont := 0 to aSerie.images.Count - 1 do
    begin
    refCycImage := aSerie.Images[cont];
    thumbsManager.addImage(refCycImage);
    end;
  sbThumbImages.EnableAutoRange;
end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.showSerieImagesStepByStep(aCyclopsImage: TCyclopsImage);
//var
//    refIm                 : TImage;
//    cont                  : Integer;
//    refBmp                : Graphics.TBitmap;
//    lastTop               : Integer;
//    refCycImage           : TCyclopsImage;
//    size: integer;
begin
  sbThumbImages.DisableAutoRange;
//  thumbsManager.clearThumbs;
//  for cont := 0 to aSerie.images.Count - 1 do
//    begin
//    refCycImage := aSerie.Images[cont];
    thumbsManager.addImage(aCyclopsImage);
    sbThumbImages.Repaint;
//    end;
  sbThumbImages.EnableAutoRange;
end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.OpenPatientStudies1Click(
  Sender: TObject);
begin
  if selectedPatient <> nil then
    showPatientStudy(selectedPatient);
end;
//------------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.OpenStudySeries1Click(Sender: TObject);
begin
  if (selectedPatient <> nil) and
     (selectedPatient.studies.Count > 0) then
```

```
    showStudySeries(selectedStudy);
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.Open2Click(Sender: TObject);
var
  viewer: TfImageViewer;
begin
            try
       viewer:= TfImageViewer.createWith(self, selectedSerie.images,selectedPatient);
       viewer.Caption := viewer.Caption + '  ( ' + sgPatient.Cells[0,sgPatient.Row] +' )';
       viewer.ShowModal;

       viewer.Free;
    except

    end;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  clearTempDir;
  Close1Click(self);
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.FormResize(Sender: TObject);
var
  h: integer;
begin
  h:= panel3.Height div 2;
  panel4.Height:= h;
  panel5.Top:= h + 1;
  panel5.Height:= h;
end;
//---------------------------------------------------------------------------
function TfCyclopsDICOMEditorMain.openDirectory(path: string): TStringList;
var
  SearchRec: TSearchRec;
  files, subFiles: TStringList;
begin
  files:= TStringList.Create;
  if FindFirst(path + '\*.dcm', faAnyFile, SearchRec) = 0 then
    begin
    files.Add(path + '\' + searchRec.Name);
    while FindNext(SearchRec) = 0 do
      files.Add(path + '\' + searchRec.Name);
    end;
  findClose(SearchRec);
  if FindFirst(path + '\*.*', faDirectory, SearchRec) = 0 then
    begin
    if (SearchRec.Name <> '.') and (SearchRec.Name <> '..') then
      begin
      subFiles:= openDirectory(path + '\' + searchRec.Name);
      files.AddStrings(subFiles);
      subFiles.Free;
      end;
    while FindNext(SearchRec) = 0 do
      if (SearchRec.Name <> '.') and
        (SearchRec.Name <> '..') and
        (DirectoryExists(path + '\' + searchRec.Name)) then
        begin
        subFiles:= openDirectory(path + '\' + searchRec.Name);
        files.AddStrings(subFiles);
        subFiles.Free;
        end;
    end;
  result:= files;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.ShowServerStatus1Click(Sender: TObject);
```

```
begin
  fServerStarter.ShowModal;
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.About1Click(Sender: TObject);
var
        aboutForm : TfAbout;
begin
        aboutForm := TfAbout.Create(self);
        aboutForm.ShowModal;
   aboutForm.free;
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.ServerOptions1Click(Sender: TObject);
//var
//  opt: TFOptions;
begin
//  opt:= TfOptions.createWith(self, searchPaths);
//  opt.ShowModal;
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.OpenQueryTool1Click(Sender: TObject);
begin
  fQueryTool.ShowModal;
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.clearTempDir;
var
  SearchRec: TSearchRec;
  path: string;
begin
  path:= ExtractFilePath(Application.ExeName) + 'tmp\';
  if FindFirst( path + '*.*', faAnyFile, SearchRec) = 0 then
    begin
    DeleteFile(path + SearchRec .Name);
    while FindNext(SearchRec) = 0 do
      DeleteFile(path + SearchRec.Name);
    end;
  findClose(SearchRec);
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.sgSeriesContextPopup(Sender: TObject;
  MousePos: TPoint; var Handled: Boolean);
begin
  if (selectedStudy = nil) then
    handled:= true
  else
    begin
    if selectedSerie = nil then
      selectedSerie:= selectedStudy.Series.items[sgSeries.row - 1];
    end;
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.PrintImage1Click(Sender: TObject);
begin
end;

//----------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.PrintSeries1Click(Sender: TObject);
var
  fPrint      : TfImagePrint;
  imgList   : TList;
  bmps                : array of TBitmap;
  cycImg    : TCyclopsImage;
```

```
  i                                   : integer;
  img                 : TCyclopsImage;
//  bmp                 : TBitmap;
begin
            imgList := TList.Create;
    //SetLength(imgIndexes,selectedSerie.images.Count);
    SetLength(bmps,selectedSerie.images.Count);
            for i := 0 to selectedSerie.images.Count - 1 do
    begin
            cycImg := selectedSerie.images.Items[i];
       img := TCyclopsImage.Create(nil);
       bmps[i] := cycImg.getbitmap;
       img.Picture.Bitmap := bmps[i];
                        img.copy(cycImg);
       imgList.add(img);
    end;
            fPrint:= TfImagePrint.CreateWith(Self,imgList);
            fPrint.setWith(selectedPatient);
            fPrint.ShowModal;
    //release memory

    for i := 0 to selectedSerie.images.Count - 1 do
    begin
       img := imgList.Items[i];
       //bmp := img.Picture.Bitmap;
       img.Parent := nil;
       img.Free;
//        img := nil;
       bmps[i].Free;

    end;
    fPrint.Free;
    imgList.Clear;
//    imgList := nil;
end;
//---------------------------------------------------------------------------
{procedure TfCyclopsDICOMEditorMain.showImagesInfo(aSerie: TCyclopsSeries);
var
            refIm                               : TImage;
    cont                        : Integer;
    refBmp                      : Graphics.TBitmap;
    lastTop                     : Integer;
    refCycImage         : TCyclopsImage;
    size: integer;
begin
  for cont := 0 to aSerie.images.Count - 1 do
  begin
    sgImages.Cells[0,cont + 1] := IntToStr(cont);
    sgImages.RowCount := sgImages.RowCount + 1;

  end;

end;
}
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.openwithminiviewer1Click(
  Sender: TObject);
var
       mv     : TMiniView;
       refIm   : TCyclopsImage;
begin
       refIm := (popThumbies.PopupComponent as TCyclopsImage);
       mv := TMiniView.Create(self);
       mv.im_miniView.Picture.Bitmap := refIm.Picture.Bitmap;
       mv.im_miniView.Canvas.Pen.Color := clred;
       mv.Width := refIm.Picture.Bitmap.Width;
       mv.Height := refIm.Picture.Bitmap.Height;
       mv.Show;

end;
```

```
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.openWithMiniViewer2Click(
  Sender: TObject);
begin

end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.sgImagesSelectCell(Sender: TObject;
  ACol, ARow: Integer; var CanSelect: Boolean);
begin
  if (selectedStudy <> nil) and
     (selectedStudy.series.Count > 0) then
    begin
//    selectedSerie:= selectedStudy.series.items[aRow - 1];
//    showSerieImages(selectedSerie);
//    showImagesInfo(selectedSerie);
    end;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.RecoverImageInformation1Click(
  Sender: TObject);
begin
  recoverImageInfo;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.recoverImageInfo;
var
              cycImage  : TCyclopsImage;
    cont                : Integer;
begin
  if selectedSerie <> nil then
    begin
    thumbsManager.clearThumbs;
//    selectedSerie.convertAllImages;
//    showSerieImages(selectedSerie);
    for cont := 0 to (selectedSerie.images.Count- 1) do
    begin
    //selectedSerie.convertAllImages;
       selectedSerie.convertImageAtIndex(cont);
       cycImage := selectedSerie.images.Items[cont];
                showSerieImagesStepByStep(cycImage);
    end;

    end;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.openmailler1Click(Sender: TObject);
var
  mailler: TfrmMailler;
begin

  mailler:= TfrmMailler.createWith(self, selectedSerie.images, selectedPatient);
//  viewer.Caption := viewer.Caption + '  ( ' + sgPatient.Cells[0,sgPatient.Row] +' )';
  mailler.Show;
end;
//---------------------------------------------------------------------------
procedure TfCyclopsDICOMEditorMain.sgSeriesDblClick(Sender: TObject);
begin
              Open2Click(self);
end;
//---------------------------------------------------------------------------
//Description: this method handle all kind of abnormal events occoured in the
//project
procedure TfCyclopsDICOMEditorMain.ApplicationEvents1Exception(
  Sender: TObject; E: Exception);
begin

end;
//---------------------------------------------------------------------------
end.
```

```
//------------------------------------------------------------------------------------------------------

unit ufAbout;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls;

type
  TfAbout = class(TForm)
    Image1: TImage;
    RichEdit1: TRichEdit;
    Button1: TButton;
    Panel2: TPanel;
    Image2: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fAbout: TfAbout;

implementation

{$R *.dfm}

procedure TfAbout.Button1Click(Sender: TObject);
begin
    Close;
end;

end.

//------------------------------------------------------------------------------------------------------

unit ufDirectoryFilter;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, uCyclopsDicomServer, FileCtrl,
  Menus;

type
  TfDirectoryFilter = class(TForm)
    dlbDir: TDirectoryListBox;
    dcbDriver: TDriveComboBox;
    lbSearchPaths: TListBox;
    Button2: TButton;
    BitBtn1: TBitBtn;
    Bevel2: TBevel;
    Shape5: TShape;
    Label2: TLabel;
    Panel1: TPanel;
    Image1: TImage;
    PopupMenu1: TPopupMenu;
    addselected1: TMenuItem;
    remove1: TMenuItem;
    Shape1: TShape;
    Label1: TLabel;
    procedure dcbDriverChange(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
```

```
    procedure remove1Click(Sender: TObject);
  private
    { Private declarations }
    searchPaths: TStringList;
  public
    { Public declarations }
    constructor createWith(AOwner: TComponent; paths: TStringList);
  end;

var
  fDirectoryFilter: TfDirectoryFilter;

implementation

{$R *.dfm}

{ TfOptions }

constructor TfDirectoryFilter.createWith(AOwner: TComponent; paths: TStringList);
begin
  inherited create(AOwner);
  searchPaths:= paths;
  lbSearchPaths.items.AddStrings(paths);
end;

procedure TfDirectoryFilter.dcbDriverChange(Sender: TObject);
begin
  dlbDir.Drive:= dcbDriver.Drive;
end;

procedure TfDirectoryFilter.Button1Click(Sender: TObject);
var
  item: string;
begin
  item:= dlbDir.GetItemPath(dlbDir.ItemIndex);
  lbSearchPaths.Items.Add(item);
  searchPaths.Add(item);
end;

procedure TfDirectoryFilter.BitBtn1Click(Sender: TObject);
var
  item                  : string;
  cont                  : Integer;
  strCmp    : string;
begin
  item:= dlbDir.GetItemPath(dlbDir.ItemIndex);
  for cont := 0 to searchPaths.Count- 1 do
  begin
            strCmp := searchPaths.Strings[cont];
    if strCmp = item then
            exit;
  end;
  searchPaths.Add(item);
  lbSearchPaths.Items.Add(item);
end;

procedure TfDirectoryFilter.Button2Click(Sender: TObject);
begin
  Close;
end;

procedure TfDirectoryFilter.remove1Click(Sender: TObject);
var
            cont                  : Integer;
begin
            for cont := 0 to lbSearchPaths.Items.count-1 do
  begin
                        if lbSearchPaths.Selected[cont] then
                                    lbSearchPaths.Items.Delete(cont);
      searchPaths.Delete(cont);
```

```
  end;
end;

end.

//----------------------------------------------------------------------------------------------------

unit ufOptions;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, uCyclopsDicomServer, FileCtrl,
  Menus;

type
  TfOptions = class(TForm)
    dlbDir: TDirectoryListBox;
    dcbDriver: TDriveComboBox;
    lbSearchPaths: TListBox;
    Button2: TButton;
    BitBtn1: TBitBtn;
    Bevel2: TBevel;
    Bevel1: TBevel;
    Shape5: TShape;
    Shape1: TShape;
    Label2: TLabel;
    Label1: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    cbHostName: TComboBox;
    cbPort: TComboBox;
    cbDicomDBServerName: TComboBox;
    cbServerName: TComboBox;
    Button1: TButton;
    Bevel3: TBevel;
    Panel1: TPanel;
    Image1: TImage;
    PopupMenu1: TPopupMenu;
    addselected1: TMenuItem;
    remove1: TMenuItem;
    procedure dcbDriverChange(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure remove1Click(Sender: TObject);
  private
    { Private declarations }
    searchPaths: TStringList;
  public
    { Public declarations }
    constructor createWith(AOwner: TComponent; paths: TStringList);
  end;

var
  fOptions: TfOptions;

implementation

{$R *.dfm}

{ TfOptions }

constructor TfOptions.createWith(AOwner: TComponent; paths: TStringList);
begin
```

```
  inherited create(AOwner);
  searchPaths:= paths;
  lbSearchPaths.items.AddStrings(paths);
end;

procedure TfOptions.dcbDriverChange(Sender: TObject);
begin
  dlbDir.Drive:= dcbDriver.Drive;
end;

procedure TfOptions.Button1Click(Sender: TObject);
var
  item: string;
begin
  item:= dlbDir.GetItemPath(dlbDir.ItemIndex);
  lbSearchPaths.Items.Add(item);
  searchPaths.Add(item);
  TCyclopsDICOMServer.getcurrentServer.HostName := cbHostName.Text;
  TCyclopsDICOMServer.getcurrentServer.Port := cbPort.Text;
  TCyclopsDICOMServer.getcurrentServer.DicomDBServerName := cbDicomDBServerName.Text;
  TCyclopsDICOMServer.getcurrentServer.ServerName := cbServerName.Text;
end;

procedure TfOptions.BitBtn1Click(Sender: TObject);
var
  item                  : string;
  cont                  : Integer;
  strCmp    : string;
begin
  item:= dlbDir.GetItemPath(dlbDir.ItemIndex);
  for cont := 0 to searchPaths.Count- 1 do
  begin
            strCmp := searchPaths.Strings[cont];
    if strCmp = item then
            exit;
  end;
  searchPaths.Add(item);
  lbSearchPaths.Items.Add(item);
end;

procedure TfOptions.Button2Click(Sender: TObject);
begin
  Close;
end;

procedure TfOptions.FormCreate(Sender: TObject);
begin
  cbHostName.Items.Clear;
  cbPort.Items.Clear;
  cbDicomDBServerName.Items.Clear;
  cbServerName.Items.Clear;
  cbHostName.Items.Add(TCyclopsDICOMServer.getcurrentServer.HostName);
  cbPort.Items.Add(TCyclopsDICOMServer.getcurrentServer.Port);
  cbDicomDBServerName.Items.Add(TCyclopsDICOMServer.getcurrentServer.DicomDBServerName);
  cbServerName.Items.Add(TCyclopsDICOMServer.getcurrentServer.ServerName);
end;

procedure TfOptions.remove1Click(Sender: TObject);
var
            cont                  : Integer;
begin
            for cont := 0 to lbSearchPaths.Items.count-1 do
  begin
                        if lbSearchPaths.Selected[cont] then
                                    lbSearchPaths.Items.Delete(cont);
     searchPaths.Delete(cont);
  end;
end;

end.
```

```
//------------------------------------------------------------------------------------------------------------

unit UfprintImages;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, uCyclopsImagesPrinter, uCyclopsPatient, uCyclopsImage,
  ExtCtrls,uCyclopsImageList;

type
  TfImagePrint = class(TForm)
    mText: TMemo;
    Label1: TLabel;
    sb1row: TSpeedButton;
    sb2row: TSpeedButton;
    sb3row: TSpeedButton;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    PrinterSetupDialog1: TPrinterSetupDialog;
    scbThumbs: TScrollBox;
    imPreview: TImage;
    BitBtn4: TBitBtn;
    btnClose: TButton;
    Panel1: TPanel;
    spbpage1: TSpeedButton;
    spbpage2: TSpeedButton;
    spbpage3: TSpeedButton;
    spbpage4: TSpeedButton;
    spbpage5: TSpeedButton;
    spbpage6: TSpeedButton;
    Label2: TLabel;
    spbpage7: TSpeedButton;
    spbpage8: TSpeedButton;
    spbpage9: TSpeedButton;
    procedure sb1rowClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure sb2rowClick(Sender: TObject);
    procedure sb3rowClick(Sender: TObject);
    procedure sb4rowClick(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure imPreviewDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure imPreviewDragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure BitBtn4Click(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure spbpage1Click(Sender: TObject);
    procedure spbpage2Click(Sender: TObject);
    procedure spbpage3Click(Sender: TObject);
    procedure spbpage4Click(Sender: TObject);
    procedure spbpage5Click(Sender: TObject);
    procedure spbpage6Click(Sender: TObject);
    procedure spbpage7Click(Sender: TObject);
    procedure spbpage8Click(Sender: TObject);
    procedure spbpage9Click(Sender: TObject);
  private
    imagesPerRow        : integer;
    cyclopsPrinter      : TCyclopsImagesPrinter;
    images              : TList;
    selectedPatient     : TCyclopsPatient;
    imagesforPage       : integer;
//    imagesPos1          : array [1..1] of integer;
//    imagespos4          : array [1..4] of integer;
//    imagesPos9          : array [1..9] of integer;
```

```
    ////////////////////////////////////////////////
    pos9                                    : array[1..9,1..9] of integer;
    pos4                                    : array[1..9,1..4] of integer;
    pos1                                    : array[1..9,1..1] of integer;
    ////////////////////////////////////////////////
    currentPage                             : integer;
    usedPages                               : array[1..9] of boolean;

    procedure       drawPreview();
    procedure       clearPos;
    function prepareImageList: TList;
    procedure       drawCurrentPage;
  public
    procedure       setWith(patient: TCyclopsPatient);
    procedure       showThumbs;
    constructor     CreateWith(AOwner : TComponent; im glist : TList);
  end;

var
  fImagePrint: TfImagePrint;

implementation

{$R *.dfm}
//---------------------------------------------------------------------------
constructor TfImagePrint.CreateWith(AOwner : TComponent; imglist : TList);
var
        i: integer ;
begin
inherited create(AOwner);
        imagesPerRow:= 1;
        cyclopsPrinter:= TCyclopsImagesPrinter.create(self);
        imagesforPage := 1;

    //creates the list that will contain all images.
        images := imgList;

    //display imagens on the sbImages
    showThumbs;

    currentPage := 1;
    for i := 1 to 9 do //means that no pages are yet used
        usedPages[i] := false;
end;
//---------------------------------------------------------------------------
procedure TfImagePrint.sb1rowClick(Sender: TObject);
begin
    imagesforPage := 1;
        imagesPerRow:= 1;
        drawPreview;
        sb1row.Flat:= true;
        sb2row.Flat:= false;
        sb3row.Flat:= false;
    clearPos;
end;
//---------------------------------------------------------------------------
procedure TfImagePrint.FormCr eate(Sender: TObject);
begin
  imagesPerRow:= 1;
  cyclopsPrinter:= TCyclopsImagesPrinter.create(self);
        imagesforPage := 1;
end;
//---------------------------------------------------------------------------
procedure TfImagePrint.sb2rowClick(Sender: TObject);
begin
        imagesforPage := 4;
        drawPreview;

        imagesPerRow:= 2;
        sb1row.Flat:= false;
```

```
            sb2row.Flat:= true;
            sb3row.Flat:= false;
   clearPos;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.sb3rowClick(Sender: TObject);
begin
   imagesPerRow:= 3;
   imagesforPage := 9;
   drawPreview;
   sb1row.Flat:= false;
   sb2row.Flat:= false;
   sb3row.Flat:= true;
   clearPos;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.sb4rowClick(Sender: TObject);
begin
  imagesPerRow:= 4;
  sb1row.Flat:= false;
  sb2row.Flat:= false;
  sb3row.Flat:= false;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.BitBtn3Click(Sender: TObject);
var
  fieldNames, fieldValues: TStringList;
  img: TCyclopsImage;
  text: string;
  i: integer;
begin
  text:= '';
  for i:= 0 to mText.Lines.Count- 1 do
    text:= text + mText.Lines.Strings[i] + #13;

  img:= images.first;
  fieldNames := TStringList.Create;
  fieldValues := TStringList.Create;
  fieldNames.Add('Name');
  fieldValues.Add(selectedPatient.data.patientsName);
  fieldNames.Add('Birth Date');
  fieldValues.Add(selectedPatient.data.patientsBirthDate);

  cyclopsPrinter.ImagesPerLine:= Self.imagesPerRow;

  cyclopsPrinter.writeImageReport(text, fieldNames, fieldValues, prepareImageList);

end;
//------------------------------------------------------------------------------
procedure TfImagePrint.setWith(patient: TCyclopsPatient);
begin
  selectedPatient:= patient;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.BitBtn2Click(Sender: TObject);
var
  fieldNames, fieldValues: TStringList;
  img: TCyclopsImage;
  text: string;
  i: integer;
begin
  text:= '';
  for i:= 0 to mText.Lines.Count- 1 do
    text:= text + mText.Lines.Strings[i] + #13;
  img:= images.first;
  fieldNames := TStringList.Create;
  fieldValues := TStringList.Create;
  fieldNames.Add('Name');
  fieldValues.Add(selectedPatient.data.patientsName);
  fieldNames.Add('Birth Date');
```

```
  fieldValues.Add(selectedPatient.data.patientsBirthDate);
  cyclopsPrinter.ImagesPerLine:= Self.imagesPerRow;
  cyclopsPrinter.previewImageReport(text, fieldNames, fieldValues, prepareImageList);
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.BitBtn1Click(Sender: TObject);
begin
  PrinterSetupDialog1.Execute;
end;
//------------------------------------------------------------------------------
//description : show current thumbs
procedure TfImagePrint.showThumbs;
var
        image                   :TCyclopsImage;
    count               : Integer;
    offSet              : Integer;
    vertBack            : Integer;
begin
        offSet := 4;
    vertBack := scbThumbs.VertScrollBar.Size;
        for count := 0 to images.Count - 1 do
    begin
        Image := images.Items[count];
      Image.Top := offSet;
      Image.Left := 4;
      Image.Width := 90;
      Image.Height := 90;
      Image.Stretch := True;
      offSet := offSet + Image.Height + 4;
      Image.DragMode := dmAutomatic;
      Image.ShowHint := true;
      Image.Parent := scbThumbs;
    end;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.FormShow(Sender: TObject);
begin
        showThumbs;
    drawPreview;
end;

//------------------------------------------------------------------------------
procedure TfImagePrint.imPreviewDragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TCyclopsImage;
end;

//------------------------------------------------------------------------------
procedure TfImagePrint.imPreviewDragDrop(Sender, Source: TObject; X,
  Y:Integer);
var
        imNum    : integer;
    str                 : string;
begin
        imPreview.Canvas.Pen.Color := clRed;
    usedPages[currentPage] := true;

    imNum := (Source as TCyclopsImage).imageNumber;
    if imNum < 10 then
        str := '0' + IntToStr(imNum)
    else
       str := IntToStr(imNum);
        //if (Sender is TImage) and (Source is TCyclopsImage) then
        //begin
                if imagesforPage = 1 then
                begin
                        imPreview.Canvas.StretchDraw(rect(10,90,227,325),
     (Source as TCyclopsImage).Picture.Graphic);
                        imPreview.Canvas.TextOut(217,315, str);
```

```
Pos1[currentPage,1] := (Source as TCyclopsImage).imageNumber;
                end;
                if imagesforpage = 4 then
                begin
                        if (x > 10) and (x < 115) then //first line
                        begin
                                if (y > 95) and (y < 200) then
                                begin
    imPreview.Canvas.StretchDraw(rect(10,95,115,200),
                        (Source as TCyclopsImage).Picture.Graphic);
                                        imPreview.Canvas.TextOut(105,190,str);
     Pos4[currentPage,1] := (Source as TCyclopsImage).imageNumber;
                                end;

                                if (y > 205) and (y < 310) then
                                begin
                                        imPreview.Canvas.StretchDraw(rect(10,205,115,310),
                        (Source as TCyclopsImage).Picture.Graphic);
                                        imPreview.Canvas.TextOut(105,300,str);
     Pos4[currentPage,3] := (Source as TCyclopsImage).imageNumber;
                                end;
                        end;

                        if (x > 120) and (x < 225) then //second line
                        begin
                                if (y > 95) and (y < 200) then
                                begin
                                        imPreview.Canvas.StretchDraw(rect(120,95,225,200),
                        (Source as TCyclopsImage).Picture.Graphic);
     imPreview.Canvas.TextOut(215,190,str);
     Pos4[currentPage,2] := (Source as TCyclopsImage).imageNumber;
                                end;

                                if (y > 205) and (y < 310) then
                                begin
                                        imPreview.Canvas.StretchDraw(rect(120,205,225,310),
                        (Source as TCyclopsImage).Picture.Graphic);
     imPreview.Canvas.TextOut(215,300,str);
                                        Pos4[currentPage,4] := (Source as TCyclopsImage).imageNumber;
                                end;
                        end;

                end;

                if imagesforpage = 9 then
                begin
    if (x > 9) and (x < 79) then
begin
    if (y > 94) and (y < 164) then
   begin
                                        imPreview.Canvas.StretchDraw(rect(9,94,79,164),
                (Source as TCyclopsImage).Picture.Graphic);
                                        imPreview.Canvas.TextOut(69,154,str);
      pos9[currentPage,1] := (Source as TCyclopsImage).imageNumber;
   end;
   if (y > 170) and (y < 240) then
   begin
                                        imPreview.Canvas.StretchDraw(rect(9,170,79,240),
                (Source as TCyclopsImage).Picture.Graphic);
                                        imPreview.Canvas.TextOut(69,230,str);
      pos9[currentPage,4] := (Source as TCyclopsImage).imageNumber;
   end;
   if (y > 248) and (y < 318) then
   begin
                                        imPreview.Canvas.StretchDraw(rect(9,248,79,318),
                (Source as TCyclopsImage).Picture.Graphic);
                                        imPreview.Canvas.TextOut(69,308,str);
      pos9[currentPage,7] := (Source as TCyclopsImage).imageNumber;
   end;
end;
```

```
            if (x > 83) and (x < 153) then
        begin
            if (y > 94) and (y < 164) then
            begin
                                                    imPreview.Canvas.StretchDraw(rect(83,94,153,164),
                        (Source as TCyclopsImage).Picture.Graphic);
                                                    imPreview.Canvas.TextOut(143,154,str);
              pos9[currentPage,2] := (Source as TCyclopsImage).imageNumber;

            end;
            if (y > 170) and (y < 240) then
            begin
                                                    imPreview.Canvas.StretchDraw(rect(83,170,153,240),
                        (Source as TCyclopsImage).Picture.Graphic);
                                                    imPreview.Canvas.TextOut(143,230,str);
              pos9[currentPage,5] := (Source as TCyclopsImage).imageNumber;
            end;
            if (y > 248) and (y < 318) then
            begin
                                                    imPreview.Canvas.StretchDraw(rect(83,248,153,318),
                        (Source as TCyclopsImage).Picture.Graphic);
                                                    imPreview.Canvas.TextOut(143,308,str);
              pos9[currentPage,8] := (Source as TCyclopsImage).imageNumber;
            end;
        end;

            if (x > 157) and (x < 227) then
        begin
            if (y > 94) and (y < 164) then
            begin
                                                    imPreview.Canvas.StretchDraw(rect(157,94,227,164),
                        (Source as TCyclopsImage).Picture.Graphic);
                                                    imPreview.Canvas.TextOut(217,154,str);
              pos9[currentPage,3] := (Source as TCyclopsImage).imageNumber;
            end;
            if (y > 170) and (y < 240) then
            begin
                                                    imPreview.Canvas.StretchDraw(rect(157,170,227,240),
                        (Source as TCyclopsImage).Picture.Graphic);
                                                    imPreview.Canvas.TextOut(217,230,str);
              pos9[currentPage,6] := (Source as TCyclopsImage).imageNumber;
            end;
            if (y > 248) and (y < 318) then
            begin
                                                    imPreview.Canvas.StretchDraw(rect(157,248,227,318),
                        (Source as TCy clopsImage).Picture.Graphic);
                                                    imPreview.Canvas.TextOut(217,308,str);
              pos9[currentPage,9] := (Source as TCyclopsImage).imageNumber;
            end;
        end;


                        end;
             //end;
end;

//-----------------------------------------------------------------------------
procedure TfImagePrint.drawPreview;
var
             bmp : TBitmap;
begin
  imPreview.Canvas.Pen.Color := clWhite;
  imPreview.Canvas.Rectangle(0,0,235,333);
  imPreview.Canvas.Pen.Color := clBlack;
  imPreview.Canvas.Rectangle(5,5,230,330);


  imPreview.Canvas.Rectangle(10,10,225,40);
  imPreview.Canvas.Rectangle(10,45,225,85);
```

```
  //show cyclops and clçinic logos
  bmp := TBitmap.Create;
  bmp.LoadFromFile(extractFilePath(Application.ExeName) + \Clinic.bmp');
  imPreview.Canvas.CopyRect(rect(165,10,225,40),bmp.Canvas,rect(0,0,bmp.Width,bmp.Height));
  bmp.LoadFromFile(extractFilePath(Application.ExeName) + \Cyclops.bmp');
  imPreview.Canvas.CopyRect(rect(10,10,70,40),bmp.Canvas,rect(0,0,bmp.Width,bmp.Height));

  if imagesforPage = 1 then
  begin
                        imPreview.Canvas.Rectangle(10,90,227,327);
  end;

  if imagesforPage = 4 then
  begin
                        imPreview.Canvas.Rectangle(10,95,115,200);
                        imPreview.Canvas.Rectangle(120,95,225,200);
                        imPreview.Canvas.Rectangle(10,205,115,310);
                        imPre view.Canvas.Rectangle(120,205,225,310);
  end;

  if imagesforPage = 9 then
  begin
                        imPreview.Canvas.Rectangle(9,94,79,164);
                        imPreview.Canvas.Rectangle(83,94,153,164);
      imPreview.Canvas.Rectangle(157,94,227,164);

      imPreview.Canvas.Rectangle(9,170,79,240);
      imPreview.Canvas.Rectangle(83,170,153,240);
      imPreview.Canvas.Rectangle(157,170,227,240);

      imPreview.Canvas.Rectangle(9,248,79,318);
      imPreview.Canvas.Rectangle(83,248,153,318);
      imPreview.Canvas.Rect angle(157,248,227,318);


  end;

end;
//------------------------------------------------------------------------------
procedure TfImagePrint.BitBtn4Click(Sender: TObject);
begin
        clearPos;
        drawPreview;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.btnCloseClick(Sender: TObject);
begin
        close;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.clearPos;
var
        cont        : integer;
    cont2    : integer;
begin
    for cont := 1 to 9 do
    begin
      usedPages[cont] := false; //clear used pages
            for cont2 := 1 to 9 do
            pos9[cont,cont2] := -1;
      for cont2 := 1 to 4 do
            pos4[cont,cont2] := -1;

      pos1[cont,1] := -1;
    end;


//    imagesPos1[1] := -1;
//    for cont := 1 to 4 do
```

```
//              imagespos4[cont] := -1;
//    for cont := 1 to 9 do
//              imagespos9[cont] := -1;
end;
//-------------------------------------------------------------------------------
//description : This function creates a new cyclopsImageList in the right printer
//order.
function TfImagePrint.prepareImageList: TList;
var
              list                        : TList;
    cont,i              : integer;
    contPages           : integer;
    ok          : boolean;
//    numImgs                       : integer;
    cycImg              : TCyclopsImage;
    cycImg2             : TCyclopsImage;
    bmp                           : TBitmap;
begin
    list := TList.Create;

    //to one image by page
              if imagesPerRow = 1 then
    begin
              for contPages := 1 to 9 do
              begin
          if usedPages[contPages] then //skip not used pages
          begin
             if Pos1[contPages,1] <>  -1 then
             begin
                for i := 0 to images.Count - 1 do
                begin
                   cycImg := images[i];
                   if cycImg.imageNumber = Pos1[contPages,1] then
                   begin
                      list.Add(cycImg);
                   end;
                end;
             end else
             begin
                cycImg2 := TCyclopsImage.Create(self);
                bmp := TBitmap.Create;
                bmp.LoadFromFile(ExtractFilePath(Application.ExeName) + '\blank.bmp');
                cycImg2.Picture.Bitmap.Assign(bmp);
                list.Add(cycImg2);
             end;
                                        end;
       end;
    end;
    //to for images by page
              if imagesPerRow = 2 then
    begin
              for contPages := 1 to 9 do
       begin
          if usedPages[contPages] then //skip not used pages
          begin
             for cont := 1 to 4 do
             begin
                if Pos4[contPages,cont] <> -1 then
                begin
                   for i := 0 to images.Count- 1 do
                   begin
                      cycImg := images[i];
                      if cycImg.imageNumber = Pos4[contPages,cont] then
                      begin
                         list.Add(cycImg);
                         ok := true;
                      end;
                   end;
                end else
                begin
```

```
                cycImg2 := TCyclopsImage.Create(self);
                bmp := TBitmap.Create;
                bmp.LoadFromFile(ExtractFilePath(Application.ExeName) + '\blank.bmp');
                cycImg2.Picture.Bitmap.Assign(bmp);
                list.Add(cycImg2);
              end;
            end;
                                    end;
              end;
    end;
    //to nine images by page
    if imagesPerRow = 3 then
    begin
             for contPages := 1 to 9 do
      begin
        if usedPages[contPages] then //skip not used pages
        begin
          for cont := 1 to 9 do
          begin
            ok := false;
            if Pos9[contPages,cont] <> -1 then
            begin
              for i := 0 to images.Count- 1 do
              begin
                cycImg := images[i];
                if cycImg.imageNumber = Pos9[contPages,cont] then
                begin
                  list.Add(cycImg);
                  ok := true;
                end;
              end;
            end else
            begin
              cycImg2 := TCyclopsImage.Create(self);
              bmp := TBitmap.Create;
              bmp.LoadFromFile(ExtractFilePath(Application.ExeName) + '\blank.bmp');
              cycImg2.Picture.Bitmap.Assign(bmp);
              list.Add(cycImg2);
            end;
          end;
                                    end;
                          end; //for contpages
    end;
    Result := list;

end;
//------------------------------------------------------------------------------
procedure TfImagePrint.spbpage1Click(Sender: TObject);
begin
    currentPage := 1;
    spbpage1.Flat :=true;
    spbpage2.Flat :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
    spbpage6.Flat :=false;
    spbpage7.Flat  :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;
    drawCurrentPage;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.spbpage2Click(Sender: TObject);
begin
    currentPage := 2;
    spbpage1.Flat :=false;
    spbpage2.Flat :=true;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
```

```
    spbpage6.Flat :=false;
    spbpage7.Flat :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;
            drawCurrentPage;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.spbpage3Click(Sender: TObject);
begin
   currentPage := 3;
    spbpage1.Flat :=false;
    spbpage2.Flat :=false;
    spbpage3.Flat :=true;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
    spbpage6.Flat :=false;
    spbpage7.Flat :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;

            drawCurrentPage;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.spbpage4Click(Sender: TObject);
begin
   currentPage := 4;
    spbpage1.Flat :=false;
    spbpage2.Flat :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=true;
    spbpage5.Flat :=false;
    spbpage6.Flat :=false;
    spbpage7.Flat :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;

            drawCurrentPage;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.spbpage5Click(Sender: TObject);
begin
   currentPage := 5;
    spbpage1.Flat :=false;
    spbpage2.Flat :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=true;
    spbpage6.Flat :=false;
    spbpage7.Flat :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;
            drawCurrentPage;

            drawCurrentPage;
end;
//------------------------------------------------------------------------------
procedure TfImagePrint.spbpage6Click(Sender: TObject);
begin
   currentPage := 6;
    spbpage1.Flat :=false;
    spbpage2.Flat :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
    spbpage6.Flat :=true;
    spbpage7.Flat :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;
            drawCurrentPage;
end;
```

```
//-------------------------------------------------------------------------
procedure TfImagePrint.spbpage7Click(Sender: TObject);
begin
   currentPage := 7;
    spbpage1.Flat :=false;
    spbpage2.Flat :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
    spbpage6.Flat :=false;
    spbpage7.Flat :=true;
    spbpage8.Flat :=false;
    spbpage9.Flat :=false;
    drawCurrentPage;
end;
//-------------------------------------------------------------------------
procedure TfImagePrint.spbpage8Click(Sender: TObject);
begin
   currentPage := 8;
    spbpage1.Flat :=false;
    spbpage2.Flat :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
    spbpage6.Flat :=false;
    spbpage7.Flat :=false;
    spbpage8.Flat :=true;
    spbpage9.Flat :=false;
    drawCurrentPage;
end;
//-------------------------------------------------------------------------
procedure TfImagePrint.spbpage9Click(Sender: TObject);
begin
   currentPage := 9;
    spbpage1.Flat :=false;
    spbpage2.Flat  :=false;
    spbpage3.Flat :=false;
    spbpage4.Flat :=false;
    spbpage5.Flat :=false;
    spbpage6.Flat :=false;
    spbpage7.Flat :=false;
    spbpage8.Flat :=false;
    spbpage9.Flat :=true;
    drawCurrentPage;
end;
//-------------------------------------------------------------------------
//description: This function verify the page in use em redisplay all images
//
procedure TfImagePrint.drawCurrentPage;
var
            cont,i       : integer;
   cycImg : TCyclopsImage;
begin
   ////////////////////////////////////////////////////////////
    drawPreview;
            for cont := 1 to 9 do
    begin
      if pos9[currentPage,cont] <> -1 then
      begin
         for i := 0 to images.Count- 1 do
         begin
            cycImg := images[i];
            if cycImg.imageNumber = pos9[currentPage,cont] then
            begin
               case cont of
                  1: imPreviewDragDrop(self,cycImg,15,100);
                  2:imPreviewDragDrop(self,cycImg,90,100);
                  3:imPreviewDragDrop(self,cycImg,160,100);
                  4:imPreviewDragDrop(self,cycImg,15,180);
                  5:imPreviewDragDrop(self,cycImg,90,180);
```

```
                6:imPreviewDragDrop(self,cycImg,160,180);
                7:imPreviewDragDrop(self,cycImg,15,250);
                8:imPreviewDragDrop(self,cycImg,90,250);
                9:imPreviewDragDrop(self,cycImg,160,250);
                                                    end;
            end;
        end;
                    end;
    end;

    for cont := 1 to 4 do
    begin
      if pos4[currentPage,cont] <> -1 then
      begin
        for i := 0 to images.Count- 1 do
        begin
          cycImg := images[i];
          if cycImg.imageNumber = pos4[currentPage,cont] then
          begin
            case cont of
            1: imPreviewDragDrop(self,cycImg,15,100);
            2: imPreviewDragDrop(self,cycImg,125,100);
            3: imPreviewDragDrop(self,cycImg,15,210);
            4: imPreviewDragDrop(self,cycImg,125,210);
                                                    end;
          end;
        end;
                    end;
    end;

          if pos1[currentPage,1] <> -1 then
    begin
      for i := 0 to images.Count - 1 do
      begin
        cycImg := images[i];
        if cycImg.imageNumber = pos1[currentPage,1] then
        begin
          imPreviewDragDrop(self,cycImg,15,100);
        end;
        end;
      end;

    ////////////////////////////////////////////////////////////
end;
//---------------------------------------------------------------------------
end.
//---------------------------------------------------------------------------------------------------------

unit ufQueryTool;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, ExtCtrls, Menus, StdCtrls, uCyclopsGeneralFunctions,QDialogs,
  UCyclopsFilemanager, UCyclopsPatient, UCyclopsStudy, uCyclopsSeries,
  uCyclopsLoadProgressForm, uCyclopsImage, uImageViewer, ufOptions,
  ToolWin, ComCtrls, ShellAPI, ufAbout;

type TPatient = record
  name, ID: string;
end;
//this is only a temporary solution (and a very awful one!!). MUST BE CHANGED!!!

type TStudy = record
  description, date, time, InstanceUID: string;
end;

type
  TfQueryTool = class(TForm)
```

```
    Panel2: TPanel;
    Shape1: TShape;
    Label1: TLabel;
    sgPatient: TStringGrid;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Panel4: TPanel;
    Label2: TLabel;
    Shape2: TShape;
    sgStudies: TStringGrid;
    popNetStudies: TPopupMenu;
    popNetPatients: TPopupMenu;
    N1: TMenuItem;
    clearallpatients1: TMenuItem;
    Panel1: TPanel;
    configuration1: TMenuItem;
    Button1: TButton;
    loadpatientseries1: TMenuItem;
    DownloadStudySeries1:TMenuItem;
    Panel3: TPanel;
    StatusBar1: TStatusBar;
    ePatientID: TEdit;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Close1: TMenuItem;
    N2: TMenuItem;
    EditServersConfiguration1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    About2: TMenuItem;
    Image1: TImage;
    procedure File1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure loadpatientseries1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure lo1Click(Sender: TObject);
    procedure DownloadStudySeries1Click(Sender: TObject);
    procedure About2Click(Sender: TObject);
    procedure Close1Click(Sender: TObject);
    procedure clearallpatients1Click(Sender: TObject);
  private
    hostName: string;
    serverName: string;
    serverPort: string;
  public
    constructor createWith(AOwner: TComponent; vhostName: string;
                           vserverName: string; vserverPort: string);
    procedure createQueryBatch(command: string);
              procedure parsePatientResult;
              procedure parseStudyResult;
    procedure processTagOnPatient(tagNumber, tagValue: string; var patient: TPatient);
    procedure processTagOnStudy(tagNumber, tagValue: string; var study: TStudy);
    procedure addPatient(patient: TPatient);
    procedure addStudy(study: TStudy);
    procedure createPatientCols;
    procedure createStudiesCols;
  end;

var
  fQueryTool: TfQueryTool;

implementation

{$R *.dfm}

{ TForm1 }
//---------------------------------------------------------------------------
procedure TfQueryTool.File1Click(Sender: TObject);
begin
```

```
end;
//------------------------------------------------------------------------------
procedure TfQueryTool.Button1Click(Sender: TObject);
var
  query: string;
  dir: string;
begin
  dir:= ExtractFilePath(Application.ExeName);
  query := 'c:\offis\findscu -v -aec IMAGECTN_PE -S --key 0008,0052="STUDY" --key 0010,0010="" --key 0010,0020="" +
ePatientID.Text + "";
  query := query + '  --key 0010,0020="' + sgPatient.Cells[1, sgPatient.Row] + "";
  query := query + {put server info here!!!}  ' -aet Boca pe 4006';
  query := query + ' > "' + dir + 'queryResult.cyc"';

  //create query
  createQueryBatch(query);

  //execute query
  ShellExecute(0, nil, PChar('query.bat'),nil , PChar(dir), SW_HIDE);

  //take retrieve data (queryResult.cyc) and parse it!!!!!!!!!!
  parsePatientResult;
  //remove unused files
  DeleteFile(dir + 'query.bat');
  DeleteFile(dir + 'queryResult.cyc');

end;
//------------------------------------------------------------------------------
procedure TfQueryTool.createQueryBatch(command: string);
var
  dir: string;
  refFile : TextFile;
begin
  dir:= ExtractFilePath(Application.ExeName);

  AssignFile(refFile,dir + 'query.bat');
  Rewrite(refFile);

  Writeln(refFile,command);
  CloseFile(refFile);
end;
//------------------------------------------------------------------------------
constructor TfQueryTool.createWith(AOwner: TComponent; vhostName,
  vserverName, vserverPort: string);
begin
  hostName:= vhostName;
  serverName:= vserverName;
  serverPort:= vserverPort;
  inherited create(Owner);
end;
//------------------------------------------------------------------------------
procedure TfQueryTool.parsePatientResult;
var
        sl              : TStringList;
    dir, line, tagNumber, tagValue: string;
    i, j, ind1, cont: integer;
    pat: TPatient;
    str: string;
    ok : boolean;
begin
    dir:= ExtractFilePath(Application.ExeName);
        sl := TStringList.Create;
      //waits until files close
                ok := false;
                while not ok do
                begin
                  try
          sl.LoadFromFile(dir + 'queryResult.cyc');
          ok:= true;
        except
```

```
          on E: Exception do ok:= false;
        end;
                        end;


      i:= 0;
      line:= sl.Strings[0];
      str:= copy(line, 0 , 8);
       while str <> 'RESPONSE' do
        begin
        i:= i+1;
        line:= sl.Strings[i];
       str:= copy(line, 0 , 8);
        end;
      //run until to first response


      i:= i+1; //steps response line

      while i < sl.Count do
         begin
         line:= sl.Strings[i];
        pat.name:= '';
        pat.ID:= '';
        while (copy(line, 0 , 8) <> 'RESPONSE') and
              (i < sl.Count) do
          begin
          line:= sl.Strings[i];
          if (length(line) > 0) and (line[1] = '(') then
          //if the line doesn´t start with it means it´s not a tag value line
            begin
            tagNumber:= copy(line, 0 , 11);
            ind1:= -1;
            cont:= -1;
            for j:= 11 to length(line) do
              if line[j] = '[' then
                begin
                 ind1:= j + 1;
                break;
                end;
            for j:= ind1 to length(line) do
              if line[j] = ']' then
                begin
                cont:= j - ind1;
                break;
                end;
            tagValue:= copy(line, ind1, cont);
            processTagOnPatient(tagNumber, tagValue, pat);
            end;
          i:= i+1;
          end;
        addPatient(pat);
        i:= i+1;
        end;
end;
//----------------------------------------------------------------------------
procedure TfQueryTool.processTagOnPatient(tagNumber, tagValue: string; var patient: TPatient);
var
  GN, EN: string;
begin
  GN:= copy(tagNumber, 2, 4);
  EN:= copy(tagNumber, 7, 4);
  if GN = '0010' then
    begin
    if EN = '0010' then
      patient.name := tagValue
    else
    if EN = '0020' then
      patient.ID   := tagValue;
```

```
    end;
end;
//------------------------------------------------------------------------------
procedure TfQueryTool.addPatient(patient: TPatient);
var
  i: integer;
begin
  for i:= 1 to sgPatient.rowCount - 1 do
    if sgPatient.Cells[1, i] = patient.ID then
      exit;
  sgPatient.RowCount:= sgPatient.RowCount + 1;
  sgPatient.Cells[0, sgPatient.RowCount - 1] := patient.name;
  sgPatient.Cells[1, sgPatient.RowCount - 1] := patient.ID;
end;
//------------------------------------------------------------------------------
procedure TfQueryTool.loadpatientseries1Click(Sender: TObject);
var
  query: string;
  dir: string;
begin
  dir:= ExtractFilePath(Application.ExeName);
  //this must be changed so we can put any directory we want

  //(0008,1030) - Study Description
  //(0008,0020) - Study Date
  //(0008,0030) - Study Time
  //(0020,000D) - Study Instance UID

  query := 'c:\offis\findscu -v -aec IMAGECTN_PE -S --key 0008,0052="STUDY" --key 0008,1030="" --key 0008,0020="" --key
0008,0030="" --key 0020,000D="""';
  query := query + '  --key 0010,0020="' + sgPatient.Cells[1, sgPatient.Row] + '"';
  query := query + {put server info here!!!} ' -aet Boca pe 4006';
  query := query + '> "' + dir + 'queryResult.cyc"';

  //create query
  createQueryBatch(query);

  //execute query
  repeat until fileExists(dir + 'query.bat');
  ShellExecute(0, nil, PChar('query.bat'),nil , PChar(dir), SW_HIDE);

  parseStudyResult;
  //remove unused files
  DeleteFile(dir + 'query.bat');
  DeleteFile(dir + 'queryResult.cyc');
end;
//------------------------------------------------------------------------------
procedure TfQueryTool.createPatientCols;
begin
    sgPatient.Cells[0,0] := 'Patient"s Name';
    sgPatient.Cells[1,0] := 'Patient ID';
end;
//------------------------------------------------------------------------------
procedure TfQueryTool.createStudiesCols;
begin
    sgStudies.Cells[0,0] := 'Study Description';
    sgStudies.Cells[1,0] := 'Study Date';
    sgStudies.Cells[2,0] := 'Study Time';
    sgStudies.Cells[3,0] := 'Accession Number';
    sgStudies.Cells[4,0] := 'Referring Physician"s Name';
    sgStudies.Cells[5,0] := 'Study Instance UID';
    sgStudies.Cells[6,0] := 'Name of Physician(s) Reading Study';
    sgStudies.Cells[7,0] := 'Patient"s Age';
    sgStudies.Cells[8,0] := 'Patient"s Weight';

end;
//------------------------------------------------------------------------------
procedure TfQueryTool.FormCreate(Sender: TObject);
begin
  createPatientCols;
```

```
  createStudiesCols;
end;
//----------------------------------------------------------------------------
procedure TfQueryTool.lo1Click(Sender: TObject);
begin

end;
//----------------------------------------------------------------------------
procedure TfQueryTool.parseStudyResult;
var
              sl              : TStringList;
   dir, line, tagNumber, tagValue: string;
   i, j, ind1, cont: integer;
   study: TStudy;
   str: string;
   ok: boolean;
begin
   dir:= ExtractFilePath(Application.ExeName);
            sl := TStringList.Create;
       //waits until files close
                              ok := false;
                              while not ok do
                              begin
                                try
          sl.LoadFromFile(dir + 'queryResult.cyc');
          ok:= true;
         except
          on E: Exception do  ok:= false;
         end;
                              end;
      i:= 0;
      line:= sl.Strings[0];
      str:= copy(line, 0 , 8);
       while str <> 'RESPONSE' do
        begin
        i:= i+1;
        line:= sl.Strings[i];
       str:= copy(line, 0 , 8);
       end;
      //run until to first response


      i:= i+1; //steps response line

       while i < sl.Count do
          begin
          line:= sl.Strings[i];
          study.description:= '';
          study.date := '';
          study.time := '';
          study.InstanceUID := '';
         while (copy(line, 0 , 8) <> 'RESPONSE') and
               (i < sl.Count) do
           begin
           line:= sl.Strings[i];
           if (length(line) > 0) and (line[1] = '(') then
           //if the line doesn´t start with it means it´s not a tag value line
             begin
             tagNumber:= copy(line, 0 , 11);
             ind1:= -1;
             cont:= -1;
             for j:= 11 to length(line) do
               if line[j] = '[' then
                 begin
                  ind1:= j + 1;
                 break;
                 end;
             for j:= ind1 to length(line) do
               if line[j] = ']' then
                 begin
```

```
              cont:= j - ind1;
              break;
              end;
            tagValue:= copy(line, ind1, cont);
            processTagOnStudy(tagNumber, tagValue, study);
            end;
          i:= i+1 ;
          end;
        addStudy(study);
        i:= i+1;
        end;
end;
//---------------------------------------------------------------------------
procedure TfQueryTool.addStudy(study: TStudy);
begin
  sgStudies.RowCount:= sgStudies.RowCount + 1;
  sgStudies.Cells[0, sgStudies.RowCount- 1] := study.description;
  sgStudies.Cells[1, sgStudies.RowCount- 1] := study.date;
  sgStudies.Cells[2, sgStudies.RowCount- 1] := study.time;
  sgStudies.Cells[5, sgStudies.RowCount- 1] := study.InstanceUID;
end;
//---------------------------------------------------------------------------
procedure TfQueryTool.processTagOnStudy(tagNumber, tagValue: string;
  var study: TStudy);
var
  GN, EN: string;
begin
  //(0008,1030) - Study Description
  //(0008,0020) - Study Date
  //(0008,0030) - Study Time
  //(0020,000D) - Study Instance UID

  GN:= UpperCase(copy(tagNumber, 2, 4));
  EN:= UpperCase(copy(tagNumber, 7, 4));
  if GN = '0008' then
    begin
    if EN = '0020' then
      study.date := t agValue
    else
    if EN = '0030' then
      study.time   := tagValue
    else
    if EN = '1030' then
      study.description := tagValue;
    end
  else
  if GN = '0020' then
    begin
    if EN = '000D' then
      study.InstanceUID:= tagValue;
    end;
end;
//---------------------------------------------------------------------------
procedure TfQueryTool.DownloadStudySeries1Click(Sender: TObject);
var
  query: string;
  dir: string;
begin
  dir:= ExtractFilePath(Application.ExeName);

  //this must be changed so we can put any directory we want
  //(0008,1030) - Study Description
  //(0008,0020) - Study Date
  //(0008,0030) - Study Time
  //(0020,000D) - Study Instance UID
  //
  //movescu -S -v -aet Mandibula -aec IMAGECTN_PE -aem Mandibula --key 0008,0052="SERIES"
  //--key 0020,000D="1.2.840.113619.2.22.287.1.19990420.262327" pe 4006
  //
```

```
  query := 'c:\offis\movescu -S -v -aet Mandibula -aec IMAGECTN_PE -aem Mandibula';
  query := query + ' --key 0008,0052="SERIES --key 0020,000D="';
  query := query + sgStudies.Cells[5, sgStudies.Row] + '" pe 4006';

  //create query
  createQueryBatch(query);

  //execute query
  repeat until fileExists(dir + 'query.bat');
  ShellExecute(0, nil, PChar('query.bat'),nil , PChar(dir), SW_HIDE);

  //remove unused files
  DeleteFile(dir + 'query.bat');
  DeleteFile(dir + 'queryResult.cyc');
end;
//---------------------------------------------------------------------------
procedure TfQueryTool.About2Click(Sender: TObject);
begin
    fAbout.ShowModal;
end;

procedure TfQueryTool.Close1Click(Sender: TObject);
begin
    Close;
end;

procedure TfQueryTool.clearallpatients1Click(Sender: TObject);
begin
    sgPatient.RowCount := 2;
end;

end.

//---------------------------------------------------------------------------------------------------------

unit UfrmLayout;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, ExtCtrls, ComCtrls,UCyclopsImageList,UCyclopsImage,
  Buttons,UPaintBoxEx;

type
  TfrmLayout = class(TForm)
    Bevel1: TBevel;
    btnOk: TButton;
    seImagesInARow: TSpinEdit;
    Label1: TLabel;
    sbInfo: TStatusBar;
    Panel1: TPanel;
    imPreview: TImage;
    btnTwo: TBitBtn;
    Bevel2: TBevel;
    btnTree: TBitBtn;
    btnOriginal: TBitBtn;
    btnFour: TBitBtn;
    btnOne: TBitBtn;
    btnFive: TBitBtn;
    procedure seImagesInARowChange(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnOkClick(Sender: TObject);
    procedure btnOriginalClick(Sender: TObject);
    procedure btnOneClick(Sender: TObject);
    procedure btnTwoClick(Sender: TObject);
    procedure btnTreeClick(Sender: TObject);
    procedure btnFourClick(Sender: TObject);
    procedure btnFiveClick(Sender: TObject);
  private
```

```
    imageWidth          : integer;
    imageHeight         : Integer;

//    function imagesInARow: Integer;
    procedure drawPreview;
            //function imagesInARow: Integer;
    procedure setDataOnSB(index : Integer; str : string);
  public
    _imageList          : TCyclopsImageList;
    _sbMain                     : TScrollBox;

    procedure refScrollBox(var sb : TScrollBox);
    function preferedWidthSize: Integer;

    procedure drawFittedImages;
  published
            property imageList : TCyclopsImageList
            read        _imageList
       write _imageList;
    property sbMain : TScrollBox
            read        _sbMain
       write _sbMain;

   property imagesWidth : Integer
            read        imageWidth
       write imageWidth;

    property imagesHeight : Integer
            read        imageHeight
       write   imageHeight;
end;

var
  frmLayout: TfrmLayout;

implementation

{$R *.dfm}
//------------------------------------------------------------------------------
{
Description : Draw preview of image´s position
}
procedure TfrmLayout.drawPreview;
var
            contW               : Integer;
    contH               : Integer;
    pvWidth                     : Integer;
    pvHeight            : Integer;
    offSetX : Integer;
    offSetY             : Integer;
    x1,y1,x2,y2         : Integer;
begin
            if (seImagesInARow.Value = 0) then //image in original size
            begin
       imPreview.Canvas.Pen.Color := clWhite;
       imPreview.Canvas.Rectangle(0,0,200,200);
       imPreview.Canvas.Pen.Color := clBlack;
            imPreview.Canvas.Rectangle(30,30,170,170);
       exit;
    end;


    imPreview.Canvas.Pen.Color := clWhite;
    imPreview.Canvas.Rectangle(0,0,200,200);
            imPreview.Canvas.Pen.Color := clBlack;

   pvWidth := (200 - ((seImagesInARow.Value+ 2) * 3)) div seImagesInARow.Value;
    pvHeight := pvWidth;
```

```
    offSetX := 3;
    offSetY := 3;
            for contW := 0 to seImagesInARow.value - 1 do
    begin
            for contH := 0 to seImagesInARow.value - 1 do
       begin
          x1 := offSetX + pvWidth * contW;
          y1 := offSetY + pvHeight * contH;
           X2 := X1 + pvWidth;
          y2 := Y1 + pvHeight;
          imPreview.Canvas.Rectangle(x1,y1,x2,y2);
          offSetY := offSetY + 3;
              end;
          offSetX := offSetX + 3;
          offSetY := 3;
    end;
end;
//------------------------------------------------------------------------------
//function TfrmLayout.imagesInARow: Integer;
//begin
//              result := seImagesInARow.value;
//end;
//------------------------------------------------------------------------------
{
Description : Return the prefered width to images to be fitted on the sb
}
function TfrmLayout.preferedWidthSize: Integer;
begin
    if (seImagesInARow.Value > 0 ) then
                        result := ((sbMain.Width - 61 - (seImagesInARow.Value * 2) + 2) div seImagesInARow.Value)
    else begin
            result := 512;
    end;
end;
//------------------------------------------------------------------------------
procedure TfrmLayout.refScrollBox(var sb: TScrollBox);
begin
    sbMain := sb;
end;
//------------------------------------------------------------------------------
procedure TfrmLayout.seImagesInARowChange(Sender: TObject);
begin
    drawPreview;
    drawFittedImages;
    setDataOnSB(1,IntToStr(imageWidth));
    setDataOnSB(3,IntToStr(imageHeight));
end;
//------------------------------------------------------------------------------
procedure TfrmLayout.FormActivate(Sender: TObject);
var
            refIm                   : TCyclopsImage;
begin
    //take image width and height
    refIm := imageList.Items[0];
    imageWidth          := refIm.Width;
    imageHeight         := refIm.Height;

            drawPreview;

    setDataOnSB(1,IntToStr(imageWidth));
    setDataOnSB(3,IntToStr(imageHeight));

end;
//------------------------------------------------------------------------------
procedure TfrmLayout.drawFittedImages;
var
            contW                   : Integer;
    contH               : Integer;
    offSetX : Integer;
    offSetY             : Integer;
```

```
    imageCount          : Integer;
    colCount            : Integer;
    rowCount            : Integer;
    refImage            : TCyclopsImage;
    pbref               : TPaintBoxEx;
begin
            imageCount := 0;
//    refImage := imageList.Items[0];
    imageWidth := preferedWidthSize;
    imageHeight := imageWidth;



    if (seImagesInARow.value = 0) then //im age in original size
    begin
            colCount := imageList.Count;
       rowCount := 1;
    end else
    begin
       colCount := (imageList.Count div(seImagesInARow.value));
       if (colCount < (imageList.Count / (seImagesInARow.value))) then
       inc(colCount);
       rowCount := seImagesInARow.value  - 1;
    end;


    offSetX := 2;
    offSetY := 2;
            for contH := 0 to colCount - 1 do
    begin
                        for contW := 0 to rowCount do
       begin
          if (imageCount >= imageList.Count) then
             break;

          refImage := imageList.Items[imageCount];
          refImage.Left := offSetX + (imageWidth * contW);
          refImage.Top := offSetY + (imageHeight * contH);
          refImage.Width          := imageWidth;
          refImage.Height := imageHeight;
          refImage.Stretch := true;
          pbref := imageList.PBItems(imageCount);
          pbRef.reposiciona;

                    offSetX := offSetX + 2;
          //take´s the next image
          inc(imageCount);
             end;
       offSetY := offSetY + 2;
       offSetX := 2;
    end;
    sbMain.Repaint;
end;
//-------------------------------------------------------------------------
procedure TfrmLayout.btnOkClick(Sender: TObject);
begin
    Close;
end;
//-------------------------------------------------------------------------
procedure TfrmLayout.setDataOnSB(index: Integer; str: string);
var
            par : TStatusPanel;
begin
    //refresh status bar
    par := sbInfo.Panels.Items[index];
    par.Text := str;
end;
//-------------------------------------------------------------------------
procedure TfrmLayout.btnOriginalClick(Sender: TObject);
begin
```

```
    seImagesInARow.value := 0;
end;
//---------------------------------------------------------------------------
procedure TfrmLayout.btnOneClick(Sender: TObject);
begin
    seImagesInARow.value := 1;
end;
//---------------------------------------------------------------------------
procedure TfrmLayout.btnTwoClick(Sender: TObject);
begin
    seImagesInARow.value := 2;
end;
//---------------------------------------------------------------------------
procedure TfrmLayout.btnTreeClick(Sender: TObject);
begin
    seImagesInARow.value := 3;
end;
//---------------------------------------------------------------------------
procedure TfrmLayout.btnFourClick(Sender: TObject);
begin
    seImagesInARow.value := 4;
end;
//---------------------------------------------------------------------------
procedure TfrmLayout.btnFiveClick(Sender: TObject);
begin
    seImagesInARow. value := 5;
end;
//---------------------------------------------------------------------------
end.
//---------------------------------------------------------------------------------------------------------

unit UFrmZoom;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TfrmZoom = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    BitBtn1: TBitBtn;
    image1: TPaintBox;
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmZoom: TfrmZoom;

implementation

{$R *.dfm}

procedure TfrmZoom.FormActivate(Sender: TObject);
begin
            //image1.Picture.Bitmap.Canvas.Rectangle(0,0,image1.Width,image1.Height);
end;

end.

//---------------------------------------------------------------------------------------------------------

unit ufServerStarter;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ShellAPI, ExtCtrls, ImgList, uCyclopsDICOMServer;

type
  TfServerStarter = class(TForm)
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    Panel1: TPanel;
    Image1: TImage;
    ImageList1: TImageList;
    Timer1: TTimer;
    lbl_status: TLabel;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
    serverAppHandle: THandle;
    last : integer;
  public
    { Public declarations }
  end;

var
  fServerStarter: TfServerStarter;

implementation

{$R *.dfm}
//---------------------------------------------------------------------------
procedure TfServerStarter.BitBtn1Click(Sender: TObject);
var
  path, fileName, pars: string;
begin
  path:= 'c:\offis\';
  fileName:= 'storescp.exe';
  pars:= '-aet Mandibula 4006';
  serverApphandle:= ShellExecute(self.handle, nil, PChar(FileName), PChar(pars), PChar(path), SW_HIDE);
  Timer1.Enabled := true;
  lbl_status.Caption:= 'Server is active';
end;
//---------------------------------------------------------------------------
procedure TfServerStarter.BitBtn2Click(Sender: TObject);
begin
  CloseHandle(serverAppHandle);
  Timer1.Enabled := false;
  lbl_status.Caption:= 'Server is inactive';
end;
//---------------------------------------------------------------------------
procedure TfServerStarter.Timer1Timer(Sender: TObject);
begin
    if last = 1 then
    begin
       ImageList1.GetBitmap(0,Image1.Picture.Bitmap);
       last := 0;
    end else
    begin
                ImageList1.GetBitmap(1,Image1.Picture.Bitmap);
       last := 1;
    end;
    Image1.Refresh;
end;
//---------------------------------------------------------------------------
procedure TfServerStarter.FormCreate(Sender: TObject);
begin
              last := 0;
```

```
end;

end.

//----------------------------------------------------------------------------------------------------

(*************************************************************************************
CLASS   : TfImageViewer
Author  :       Daniel Duarte Abdala (Caju)
Date    :       09/10/2001

Description     :       This class implements all graphic manipulation tools.
        The object "sbImages" displays all imagens contaned in "imageList". ImageList
   (see uCyclopsImageList).

Atributes :

        imageList               : list with all images
   lastTop                      : used in positioning the images on sbImages
   lastLeft                     : used in positioning the images on sbImages
   imagesWidth                  : image width
   imagesHeight                 : image height
   imagePos                     : image position in a row (initialize in 1)
        fImageViewer            : ref to form interface
   pnlZoolIsDraging  : flag indicating if zoom tool is moving
        pnlDrawingDragin        : flag indicating if drawing is moving
   pnlFlipDraging               : flag indicating if fliping tool is moving
   pnlZoomWindowDraging: flag indicating if zoom window is moving


Methods :
   name : createWith(AOwner: TComponent; cycImagesList: TList);
   description : used to initialize the most interface variables

   name : displayImages;
   description : main method to display images on sbImages

   name : calcImagePosition(var top: integer; var left : integer);
   description : calculates the next image position

   name : imagesinARow: integer;
   description : inform how many images must be acomodate in a row (at most 1)

   name : currentImage(x,y : integer): integer;
   description : this method return the index (in imageList) of the image who
   contais the specified point.
   related errors: this method has a problem to identify images that are scrolled

   name : showOnSBCurrentTool(t : TDrawingTool);
   description: shows on the status bar the current tool selected

   name : showOnSBNumImages(num : integer);
   description: Show on status bar the total number of loaded images

   name : showOnSbCurrentImage(num : Integer);
   description: shows on status bar the current images above de mouse pointer

   name : openImagesList(imgs: TList);
   description     :

Related Errors  :


Need to by Done :


*************************************************************************************)
unit uImageViewer;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls,  StdCtrls, Menus,Math,Clipbrd,
  UCyclopsImage,UPaintBoxEx,UCyclopsImageList,UCyclopsPatient,
  VectorGraphicsNodeLibrary, Buttons, ToolWin, Spin, ImgList,UfrmLayout,
  ufAbout, UFrmZoom, QComCtrls, ComCtrls, uCyclopsImagesPrinter, uFPrintImages,
  uMailler;

type
  TfImageViewer = class(TForm)
    sbInformations: TStatusBar;
    popImage: TPopupMenu;
    openMiniBrowser1: TMenuItem;
    N4: TMenuItem;
    clearallDrawings1: TMenuItem;
    clearlast1: TMenuItem;
    N5: TMenuItem;
    zoon1: TMenuItem;
    N101: TMenuItem;
    N251: TMenuItem;
    N501: TMenuItem;
    N1001: TMenuItem;
    N2001: TMenuItem;
    N4001: TMenuItem;
    savetoclipboard1: TMenuItem;
    teste1: TMenuItem;
    imListToolBar: TImageList;
    ColorDialog1: TColorDialog;
    MainMenu1: TMainMenu;
    edit1: TMenuItem;
    selectallimages1: TMenuItem;
    invertselection1: TMenuItem;
    copytoclipboard1: TMenuItem;
    help1: TMenuItem;
    about1: TMenuItem;
    N7: TMenuItem;
    Panel1: TPanel;
    sbImages: TScrollBox;
    pnlFlip: TPanel;
    Shape4: TShape;
    btnFlipClose: TSpeedButton;
    Panel8: TPanel;
    SaveDialog1: TSaveDialog;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    Bevel2: TBevel;
    rbApplyAll: TRadioButton;
    RadioButton2: TRadioButton;
    Label7: TLabel;
    pnlPilot: TPanel;
    Shape7: TShape;
    btnPilotClose: TSpeedButton;
    Label12: TLabel;
    Panel7: TPanel;
    imPilot: TImage;
    Bevel4: TBevel;
    rbCoronal: TRadioButton;
    rbSagital: TRadioButton;
    deselectallImages1: TMenuItem;
    ools1: TMenuItem;
    PrinterTool1: TMenuItem;
    Mailer1: TMenuItem;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
```

```
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ToolButton7: TToolButton;
    cboMagLevel: TComboBox;
    edtMagWidth: TEdit;
    ToolButton8: TToolButton;
    ToolButton9: TToolButton;
    PopupMenu1: TPopupMenu;
    mnuLInha: TMenuItem;
    ToolButton10: TToolButton;
    circulo1: TMenuItem;
    retngulo1: TMenuItem;
    ToolButton11: TToolButton;
    ToolButton12: TToolButton;
    ToolButton13: TToolButton;
    ToolButton14: TToolButton;
    edtWindowCenter: TEdit;
    edtWindowWidth: TEdit;
    ToolButton15: TToolButton;
    ToolButton16: TToolButton;
    ToolButton17: TToolButton;
    N1: TMenuItem;
    borracha1: TMenuItem;
    texto1: TMenuItem;
    N2: TMenuItem;
    cor1: TMenuItem;
    ToolButton18: TToolButton;
    ToolButton19: TToolButton;

    procedure openimagefile1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormActivate(Sender: TObject);
    procedure selectallimages1Click(Sender: TObject);
    procedure btnFlipTopClick(Sender: TObject);
    procedure btnFlipBotonClick(Sender: TObject);
    procedure btnFlipRightClick(Sender: TObject);
    procedure btnFlipLeftClick(Sender: TObject);
    procedure btnTextClick(Sender: TObject);
    procedure btnArrowClick(Sender: TObject);
    procedure btnLineClick(Sender: TObject);
    procedure btnCircleClick(Sender: TObject);
    procedure btnSquareClick(Sender: TObject);
    procedure close1Click(Sender: TObject);
    procedure exit1Click(Sender: TObject);
    procedure btnColorClick(Sender: TObject);
    procedure showpalette1Click(Sender: TObject);
    procedure btnEraseClick(Sender: TObject);
    procedure btnZoomClick(Sender: TObject);
    procedure imagesInARow1Click(Sender: TObject);
    procedure invertselection1Click(Sender: TObject);
    procedure teste1Click(Sender: TObject);
    procedure btnPositioningClick(Sender: TObject);
    procedure Shape2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape2MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ckbShowMagnifierClick(Sender: TObject);
    procedure Shape3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape3MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btnFlipCloseClick(Sender: TObject);
    procedure btnFlipingClick(Sender: TObject);
    procedure Shape4MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape4MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Shape4MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
```

```
  procedure btnPaletteCloseClick(Sender: TObject);
  procedure Shape5MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Shape5MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure cboMagLevelChange(Sender: TObject);
  procedure edtMagWidthChange(Sender: TObject);
  procedure btnWindowClick(Sender: TObject);
  procedure Shape6MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Shape6MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Shape6MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure btnWindowCloseClick(Sender: TObject);
  procedure btnPilotClick(Sender: TObject);
  procedure btnPilotCloseClick(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure viewzoompaltte1Click(Sender: TObject);
  procedure viewdrawingpalette1Click(Sender: TObject);
  procedure viewflippalette1Click(Sender: TObject);
  procedure about1Click(Sender: TObject);
  procedure copytoclipboard1Click(Sender: TObject);
  procedure deselectallImages1Click(Sender: TObject);
  procedure viewImagePositionigtool1Click(Sender: TObject);
  procedure viewpilotpalette1Click(Sender: TObject);
  procedure viewwindowpalette1Click(Sender: TObject);
  procedure Shape7MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Shape7MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Shape7MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure ools1Click(Sender: TObject);
  procedure PrinterTool1Click(Sender: TObject);
  procedure Mailer1Click(Sender: TObject);
  procedure ToolButton14Click(Sender: TObject);
  procedure mnuLInhaClick(Sender: TObject);
  procedure circulo1Click(Sender: TObject);
  procedure retngulo1Click(Sender: TObject);
  procedure texto1Click(Sender: TObject);
  procedure borracha1Click(Sender: TObject);
  procedure ToolButton9Click(Sender: TObject);
  procedure ToolButton7Click(Sender: TObject);
  procedure ToolButton5Click(Sender: TObject);
  procedure ToolButton4Click(Sender: TObject);
  procedure ToolButton18Click(Sender: TObject);
  procedure ToolButton17Click(Sender: TObject);
  procedure cor1Click(Sender: TObject);
  procedure FormShow(Sender: TObject);
private
    imageList  : TCyclopsImageList;
  lastTop      : integer;
  lastLeft   : integer;
  imagesWidth  : integer;
  imagesHeight : integer;
  imagePos     : integer; //image position in row
  bmps       : array of TBitmap;
  selectedPatient      : TCyclopsPatient;

  pnlZoolIsDraging  : Boolean;
    pnlDrawingDragin   : Boolean;
  pnlFlipDraging       : Boolean;
  pnlZoomWindowDraging: Boolean;
  pnlWindowDraging     : Boolean;
  pnlPilotDraging      : Boolean;
  origX        : integer;
  origY        : integer;
public
  constructor createWith(AOwner: TComponent; cycImagesList: TList;aPatient : TCyclopsPatient);
```

```
    procedure displayImages;
            procedure calcImagePosition(var top: integer; var left : integer);
    function imagesinARow: integer;
    function currentImage(x,y : integer): integer;

    procedure showOnSBCurrentTool(t : TDrawingTool);
    procedure showOnSBNumImages(num : integer);
    procedure showOnSbCurrentImage(num : Integer);
    procedure openImagesList(imgs: TList);

    procedure clearTempDir;
  end;

var
  fImageViewer: TfImageViewer;

implementation

uses Types;

{$R *.dfm}
//------------------------------------------------------------------------------
//                                                              CREATED METHODS
//------------------------------------------------------------------------------
function TfImageViewer.imagesinARow: integer;
var
        imsInRow : Integer;
begin
        imsInRow := round(sbImages.Width / imagesWidth);
    if (imsInRow < (sbImages.Width / imagesWidth)) then
        inc(imsInRow);

    result := imsInRow;
end;
//------------------------------------------------------------------------------
procedure TfImageViewer.displayImages;
var
    imIndex         : integer;  // image number at imageList
    refImage        : TCyclopsImage;
begin
        //initialize vars used to calculate image presentation position
    lastTop := 2;
    lastLeft := 2;
    imagePos := 1;

    // def image´s width and height
    refImage := imageList.Items[0];
    imagesWidth := refImage.Picture.Width;
    imagesHeight := refImage.Picture.Height;

    for imIndex := 0 to imageList.Count - 1  do
    begin

            //add image to sbImages
      refImage                  := imageList.Items[imIndex];
      refImage.Top    := lastTop;
      refImage.Left   := lastLeft;
      refImage.Width  := refImage.Picture.Width;
      refImage.Height := refImage.Picture.Height;
      refImage.Parent := sbImages;

      //reposition of paintBoxEx
      imageList.prepare(imIndex, sbInformations.Panels[5],sbInformations.Panels[7],
                                                edtWindowWidth,edtWindowCenter);

      //calc next image position
            calcImagePosition(lastTop,lastLeft);

                  //show on Status bar the number of opened images
      showOnSBNumImages(imageList.Count);
```

```
            end; //end for

end;
//------------------------------------------------------------------------------
procedure TfImageViewer.calcImagePosition(var top: integer; var left : integer);
begin
            //calc number of images in a row
    inc(imagePos);

   //calc the place´s image on the row
   if (imagePos <= imagesinARow) then
    begin
            left := left + imagesWidth + 2;
   end else
    begin
      top := top + imagesHeight + 2;
            imagePos := 1;
      left := 2;
   end;
end;
//------------------------------------------------------------------------------
//                                                                  INTERFACE METHODS
//------------------------------------------------------------------------------
procedure TfImageViewer.openimagefile1Click(Sender: TObject);
begin
end;
//------------------------------------------------------------------------------
procedure TfImageViewer.FormClose(Sender: TObject; var Action: TCloseAction);
var
            i           : integer;
//    img : TCyclopsImage;
begin

            imageList.Clear;
            for i := 0 to Length(bmps) - 1 do
   begin
            //img := imageList.Items[i];
      //img.Parent := nil;
      //img.Free;
      //img := nil;
            bmps[i].Free;
   end;

      imageList.Free;
      imageList := nil;

      clearTempDir;

end;
//------------------------------------------------------------------------------
procedure TfImageViewer.FormActivate(Sender: TObject);
begin
            //initialize vars used to calculate image presentation position
    imagePos := 1;

   pnlZoolIsDraging  := false;
   pnlDrawingDragin := false;
   pnlFlipDraging     := false;
   pnlZoomWindowDraging:= false;
   pnlWindowDraging          := false;
   pnlPilotDraging           := false;
end;
//------------------------------------------------------------------------------
procedure TfImageViewer.selectallimages1Click(Sender: TObject);
var
            im                  : TPaintBoxEx;
   cont     : integer;
begin
            for cont := 0 to imageList.Count- 1 do
   begin
```

```
      im := imageList.PBItems(cont);
                  im.beBordered := true;
   end;
end;
//------------------------------------------------------------------------------
procedure TfImageViewer.btnFlipTopClick(Sender: TObject);
begin
      if rbapplyAll.Checked then
                  imageList.flip(Ottop,true)
   else
                  imageList.flip(Ottop,false);
end;
//------------------------------------------------------------------------------

procedure TfImageViewer.btnFlipBotonClick(Sender: TObject);
begin
      if rbapplyAll.Checked then
                  imageList.flip(Otbottom, true)
   else
                  imageList.flip(Otbottom, false);
end;
//------------------------------------------------------------------------------

procedure TfImageViewer.btnFlipRightClick(Sender: TObject);
begin
      if rbapplyAll.Checked then
                  imageList.flip(Otright,true)
   else
                  imageList.flip(Otright,false);
end;
//------------------------------------------------------------------------------

procedure TfImageViewer.btnFlipLeftClick(Sender: TObject);
begin
      if rbapplyAll.Checked then
                  imageList.flip(Otleft, true)
   else
      imageList.flip(Otleft, false);
end;
//------------------------------------------------------------------------------
procedure TfImageViewer.btnTextClick(Sender: TObject);
begin
    imageList.setTool(dtTextTool);
    showOnSBCurrentTool(dtTextTool);

end;
//------------------------------------------------------------------------------

procedure TfImageViewer.showOnSBCurrentTool(t: TDrawingTool);
var
      pan                  : TStatusPanel;
   str                  : string;
begin
   pan := sbInformations.Panels.Items[1];

   case t of
     dtNone                        :        str := 'None';
      dtSelectTool    : str := 'Select';
      dtRectangleTool:        str := 'Rectangle';
      dtCircleTool     :        str := 'circle';
      dtLineTool                  :        str := 'Line';
      dtTextTool                  :        str := 'Text';
      dtMagnify                  :        str := 'Magnify';
      dtWindowTool  :        str := 'Window';
            end;
   pan.Text := str;


end;
//------------------------------------------------------------------------------
```

```
procedure TfImageViewer.btnArrowClick(Sender: TObject);
begin
    imageList.setTool(dtSelectTool);
    showOnSBCurrentTool(dtSelectTool);

end;
//------------------------------------------------------------------------------
procedure TfImageViewer.btnLineClick(Sender: TObject);
begin
    imageList.setTool(dtLineTool);
    showOnSBCurrentTool(dtLineTool);

end;
//------------------------------------------------------------------------------
procedure TfImageViewer.btnCircleClick(Sender: TObject);
begin
    imageList.setTool(dtCircleTool);
    showOnSBCurrentTool(dtCircleTool);

end;
//------------------------------------------------------------------------------

procedure TfImageViewer.btnSquareClick(Sender: TObject);
begin
    imageList.setTool(dtRectangleTool);
    showOnSBCurrentTool(dtRectangleTool);

end;
//------------------------------------------------------------------------------

procedure TfImageViewer.showOnSBNumImages(num: integer);
var
        pan                     : TStatusPanel;
begin
    pan := sbInformations.Panels.Items[3];
    pan.Text := IntToStr(num);
end;
//------------------------------------------------------------------------------
//
//Description:          close all images
procedure TfImageViewer.close1Click(Sender: TObject);
begin
        //clears the image list
//    imageList.Clear;
    //clear the image counter
        showOnSBNumImages(0);

end;
//------------------------------------------------------------------------------
procedure TfImageViewer.exit1Click(Sender: TObject);
begin
        Close;


end;
//------------------------------------------------------------------------------
procedure TfImageViewer.btnColorClick(Sender: TObject);
var
        ret                     :       boolean;
begin
    ret := ColorDialog1.Execute;

    if (ret) then
    begin
//          shpColor.Brush.Color := ColorDialog1.Color;
       imageList.setColor(ColorDialog1.Color);
    end;

end;
//------------------------------------------------------------------------------
```

```
procedure TfImageViewer.showpalette1Click(Sender: TObject);
begin
{           if (showpalette1.Checked) then
   begin
           showpalette1.Checked := false;
      pnlPalette.Visible := false
   end else
   begin
           showpalette1.Checked := true;
      pnlPalette.Visible := true;
   end;
}
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.btnEraseClick(Sender: TObject);
begin
           imageList.EraseSelected;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.btnZoomClick(Sender: TObject);
begin
   imageList.setMagnifyWidth(StrToInt(edtMagWidth.text));

   imageList.setTool(dtMagnify);
   showOnSBCurrentTool(dtMagnify);
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.showOnSbCurrentImage(num: Integer);
var
           pan                     :TStatusPanel;
begin
           pan := sbInformations.Panels.It ems[5];
   pan.Text := IntToStr(num);
end;
//---------------------------------------------------------------------------
function TfImageViewer.currentImage(x, y: integer): integer;
var
           cont       : integer;
   offSet  : integer;
   findX   : boolean;
   findY   : boolean;
   xCont   : integer;
   yCont   : integer;
begin
           offSet := 10; //border distance
   findX := false;
   findY := false;


           for cont := 0 to (imagesinARow - 1) do  //verifica horizontal
   begin
                   if ((X >= offSet) and (X <= offSet + imagesWidth)) then
      begin
           findX := true;
         break;
      end;
      offSet := offSet + 10 + imagesWidth;

   end;
   xCont := cont;

   if (findX) then                                    //verifica a vertical
   begin
           offSet := 10;
           for cont := 0 to (imageList.Count div imagesInARow) do
      begin
           if ((Y >= offSet) and (Y <= offSet + imagesHeight)) then
         begin
           findY := true;
          break;
```

```
            end;
            offSet := offSet + 10 + imagesHeight;
        end;
    end;
    yCont  := cont;
    if ((findX)and(findY)) then
    begin
            result := yCont * imagesinARow + XCont;
    end else
            result := 0;


end;
//---------------------------------------------------------------------------
procedure TfImageViewer.imagesInARow1Click(Sender: TObject);
begin
            if (imageList.Count > 0 ) then
    begin
        frmLayout.imageList := imageList;
        frmLayout.sbMain := sbImages;
        frmLayout.ShowModal;
    end else
    begin

    end;

end;
//---------------------------------------------------------------------------
procedure TfImageViewer.invertselection1Click(Sender: TObject);
var
            im                  : TPaintBoxEx;
    cont    : integer;
begin
            for cont := 0 to imageList.Count- 1 do
    begin
        im := imageList.PBItems(cont);
        if (im.beBordered) then
                                    im.beBordered := false
        else
                                    im.beBordered := true;
    end;

end;
//---------------------------------------------------------------------------
procedure TfImageViewer.teste1Click(Sender: TObject);
var
            sbPanel : TStatusPanel;
    x,y : integer;
begin

    sbPanel := sbInformations.Panels[5];
    x := Mouse.CursorPos.X {+ sbImages.HorzScrollBar.Position} - sbImages.Left;
    y := Mouse.CursorPos.Y {+ sbImages.VertScrollBar.Position} - sbImages.top;
    sbPanel.Text := IntToStr(imageList.currentImIndexUnderMouse(x,y));
            sbPanel := sbInformations.Panels[1];
    sbPanel.Text := intToStr(x) + '@' + intToStr(Y);
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.btnPositioningClick(Sender: TObject);
begin
            if (imageList.Count > 0 ) then
    begin
            sbImages.VertScrollBar.Position := 0;
        sbImages.HorzScrollBar.Position := 0;
        frmLayout.imageList := imageList;
        frmLayout.sbMain := sbImages;
        frmLayout.ShowModal;
        sbImages.VertScrollBar.Increment := frmLayout.imagesHeight;
        sbImages.HorzScrollBar.Increment := frmLayout.imagesWidth;
    end else
```

```
    begin

    end;

end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Shape2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
        pnlZoolIsDraging := true;
    origX := x;
    origY := y;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Shape2MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
        pnlZoolIsDraging := false;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.ckbShowMagnifierClick(Sender: TObject);
begin

end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Shape3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    pnlDrawingDragin := true;
    origX := x;
    origY := y;
end;
//---------------------------------------------------------------------------

procedure TfImageViewer.Shape3MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    pnlDrawingDragin := false;
end;
//---------------------------------------------------------------------------

procedure TfImageViewer.btnFlipCloseClick(Sender: TObject);
begin
        pnlFlip.Visible := false;
end;
//---------------------------------------------------------------------------

procedure TfImageViewer.btnFlipingClick(Sender: TObject);
begin
    pnlFlip.Visible := true;
end;
//---------------------------------------------------------------------------

procedure TfImageViewer.Shape4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    pnlFlipDraging := true;
    origX := x;
    origY := y;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Shape4MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
        if (pnlFlipDraging) then
    begin
      pnlFlip.Left := pnlFlip.Left + X - origX;
      pnlFlip.Top := pnlFlip.Top + Y - origY;
        end;
```

```
end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.Shape4MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    pnlFlipDraging := false;
end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.btnPaletteCloseClick(Sender: TObject);
begin
            showpalette1Click(self);
end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.Shape5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
            pnlZoomWindowDraging := true;
   origX := x;
   origY := y;
end;

//-------------------------------------------------------------------------------
procedure TfImageViewer.Shape5MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    pnlZoomWindowDraging := false;
end;

//-------------------------------------------------------------------------------
constructor TfImageViewer.createWith(AOwner: TComponent;
  cycImagesList: TList;aPatient : TCyclopsPatient);
var
  i                                  : integer;
  newList   : TList;
  cycImage: TCyclopsImage;
  img                : TCyclopsImage;
begin
  inherited create(AOwner);
  //creates the list that will contain all images.
  imageList := TCyclopsImageList.CreateWith(sbImages);
  //initialize vars used to calculate image presentation position
  imagePos := 1;

  pnlZoolIsDraging    := false;
  pnlDrawingDragin   := false;
  pnlFlipDraging      := false;
  pnlZoomWindowDraging:= false;

  newList:= TList.Create;
  SetLength(bmps, cycImagesList.Count);
  for i:= 0 to cycImagesList.Count - 1 do
  begin
    cycImage:= cycImagesList.items[i];
    bmps[i] := cycImage.getBitmap;
    img := TCyclopsImage.Create(nil);
    img.Picture.Bitmap:= bmps[i];
    img.copy(cycImage);
    newList.Add(img);
  end;
  imageList.Assign(newList);
  displayImages;
  //openImagesList(newList);
  newList.Free;

  selectedPatient := aPatient;
end;

//-------------------------------------------------------------------------------
procedure TfImageViewer.openImagesList(imgs: TList);
begin
```

```
    if (imgs.Count > 0 ) then //has images on the list
    begin
       //alocates images opened to imageList
       imageList.Assign(imgs);

       //display imagens on tne sbImages
       displayImages;
    end;
end;

//---------------------------------------------------------------------------
procedure TfImageViewer.cboMagLevelChange(Sender: TObject);
begin
      imageList.setMagnifyLevel(StrToInt(cboMagLevel.text));
{   if cboMagLevel.Text = '100%' then
       imageList.setMagnifyLevel(mag100);
    if cboMagLevel.Text = '200%' then
       imageList.setMagnifyLevel(mag200);
    if cboMagLevel.Text = '400%' then
       imageList.setMagnifyLevel(mag400);
}
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.edtMagWidthChange(Sender: TObject);
begin
               if edtMagWidth.text <> '' then
                  imageList.setMagnifyWidth(StrToInt(edtMagWidth.text));

end;
//---------------------------------------------------------------------------
//description : open the window tool bar
//this method to get the original center and width window and set on the tool bar
//components.
procedure TfImageViewer.btnWindowClick(Sender: TObject);
var
    imRef    : TCyclopsImage;
    spTmp    : TStatusPanel;
//    pbref    : TPaintBoxEx;
begin
    imageList.setTool(dtWindowTool);
    showOnSBCurrentTool(dtWindowTool);

    spTmp := sbInformations.Panels.Items[7]; //returns the image number

    //now, we must retreave the image index at imageList
    imRef := imageList.Items[imageList.ImageNumberToIndex(StrToInt(spTmp.Text))];
//  pnlWindow.Visible := true;
    edtWindowCenter.Text := IntToStr(imRef.WindowCenter);
    edtWindowWidth.Text := IntToStr(imRef.WindowWidth);
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Shape6MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    pnlWindowDraging := true;
    origX := x;
    origY := y;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Shape6MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
               if (pnlWindowDraging) then
    begin
//      pnlWindow.Left := pnlWindow.Left + X - origX;// - (pnlWindow.Width div 2 );
//      pnlWindow.Top := pnlWindow.Top + Y  - origY;
               end;

end;
//---------------------------------------------------------------------------
```

```
procedure TfImageViewer.Shape6MouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    pnlWindowDraging := false;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.btnWindowCloseClick(Sender: TObject);
begin
//  pnlWindow.Visible := false;
//    viewwindowpalette1.Checked := false;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.btnPilotClick(Sender: TObject);
begin
    pnlPilot.Visible := true;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.btnPilotCloseClick(Sender: TObject);
begin
    pnlPilot.Visible := false;
//    viewpilotpalette1.Checked := false;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.Button1Click(Sender: TObject);
var
    refPb    : TPaintBoxEx;
begin
                refPb :=     imageList.Items[0];
    refPb.changePalette;
end;
//---------------------------------------------------------------------------
procedure TfImageViewer.viewzoompaltte1Click(Sender: TObject);
begin
{               if viewzoompaltte1.Checked then
    begin
       viewzoompaltte1.Ch ecked := false;
//      btnZoomClose.Click;
    end else
    begin
       viewzoompaltte1.Checked := true;
                btnZoom.Click;
    end;
}end;
//---------------------------------------------------------------------------
procedure TfImageViewer.viewdrawingpalette1Click(Sender: TObject);
begin
{               if viewdrawingpalette1.Checked then
    begin
       viewdrawingpalette1.Checked := false;
//      btnDrawingClose.Click;
    end else
    begin
       viewdrawingpalette1.Checked := true;
                btnDrawing.Click;
    end;
}end;
//---------------------------------------------------------------------------
procedure TfImageViewer.viewflippalette1Click(Sender: TObject);
begin
{               if viewflippalette1.Checked then
    begin
       viewflippalette1.Checked := false;
       btnFlipClose.Click;
    end else
    begin
       viewflippalette1.Checked := true;
                btnFliping.Click;
    end;
}end;
//---------------------------------------------------------------------------
```

```
procedure TfImageViewer.about1Click(Sender: TObject);
begin
   fAbout.ShowModal;
end;
//----------------------------------------------------------------------------
procedure TfImageViewer.copytoclipboard1Click(Sender: TObject);
begin
            //verificar se tem mais de uma imagem selecionada
   //se tiv er mais de uma imagem selecionada informar que so pode ter uma imagem
   //selecionada por vez

   //senao , copiar para o clipboard


end;
//----------------------------------------------------------------------------
procedure TfImageViewer.deselectallImages1Click(Sender: TObject);
var
            im                  : TPaintBoxEx;
   cont     : integer;
begin
            for cont := 0 to imageList.Count- 1 do
   begin
      im := imageList.PBItems(cont);
                  im.beBordered := false;
   end;
end;
//----------------------------------------------------------------------------
procedure TfImageViewer.viewImagePositionigtool1Click(Sender: TObject);
begin
{           if viewImagePositionigtool1.Checked then
   begin
      viewImagePositionigtool1.Checked := false;
      //btnPositioning.Click;
   end else
   begin
      viewImagePositionigtool1.Checked := true;
            btnPositioning.Click;
   end;
}end;
//----------------------------------------------------------------------------
procedure TfImageViewer.viewpilotpalette1Click(Sender: TObject);
begin
{           if viewpilotpalette1.Checked then
   begin
      viewpilotpalette1.Checked := false;
      btnPilotClose.Click;
   end else
   begin
      viewpilotpalette1.Checked := true;
            btnPilot.Click;
   end;
}end;
//----------------------------------------------------------------------------
procedure TfImageViewer.viewwindowpalette1Click(Sender: TObject);
begin
{           if viewwindowpalette1.Checked then
   begin
      viewwindowpalette1.Checked := false;
//    btnWindowClose.Click;
   end else
   begin
      viewwindowpalette1.Checked := true;
            btnWindow.Click;
   end;

}end;
//----------------------------------------------------------------------------

procedure TfImageViewer.Shape7MouseDown(Sender: TObject;
```

```
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    pnlPilotDraging := true;
    origX := x;
    origY := y;
end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.Shape7MouseMove(Sender: TObject;
  Shift : TShiftState; X, Y: Integer);
begin
                if (pnlPilotDraging) then
    begin
       pnlPilot.Left := pnlPilot.Left + X - origX;
       pnlPilot.Top := pnlPilot.Top + Y - origY;
                end;

end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.Shape7MouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    pnlPilotDraging := false;
end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.ools1Click(Sender: TObject);
begin

end;
//-------------------------------------------------------------------------------
procedure TfImageViewer.PrinterTool1Click(Sender: TObject);
var
  fPrint      : TfImagePrint;
  imgList   : TList;
  bmps                  : array of TBitmap;
  cycImg   : TCyclopsImage;
  i                                 : integer;
  img                   : TCyclopsImage;
//  bmp                 : TBitmap;
begin
                imgList := TList.Create;
    //SetLength(imgIndexes,selectedSerie.images.Count);
    SetLength(bmps,imageList.Count);
                for i := 0 to imageList.Count - 1 do
    begin
                cycImg := imageList.Items[i];
       img := TCyclopsImage.Create(nil);
//        bmps[i] := cycImg.getbitmap;
       img.Picture.Bitmap := cycImg.Picture.Bitmap;//bmps[i];
                        img.copy(cycImg);
       imgList.add(img);
    end;
                fPrint:= TfImagePrint.CreateWith(Self,imgList);
                fPrint.setWith(selectedPatient);
                fPrint.ShowModal;
    //release memory

    for i := 0 to imageList.Count - 1 do
    begin
       img := imgList.Items[i];
       //bmp := img.Picture.Bitmap;
       img.Parent := nil;
       img.Free;
//        img := nil;
       bmps[i].Free;

    end;
    fPrint.Free;
    imgList.Clear;

end;
```

```
//------------------------------------------------------------------------------
procedure TfImageViewer.clearTempDir;
var
  SearchRec: TSearchRec;
  path: string;
begin
  path:= ExtractFilePath(Application.ExeName) + 'tmp\';
  if FindFirst( path + '*.*', faAnyFile, SearchRec) = 0 then
    begin
    DeleteFile(path + SearchRec .Name);
    while FindNext(SearchRec) = 0 do
      DeleteFile(path + SearchRec.Name);
    end;
  findClose(SearchRec);
end;

//------------------------------------------------------------------------------
procedure TfImageViewer.Mailer1Click(Sender: TObject);
var
  mailler: TfrmMailler;
  imgList   : TList;
  bmps                  : array of TBitmap;
  cycImg   : TCyclopsImage;
  i                                 : integer;
  img                   : TCyclopsImage;
//  bmp                   : TBitmap;
begin
            imgList := TList.Create;
    //SetLength(imgIndexes,selectedSerie.images.Count);
    SetLength(bmps,imageList.Count);
            for i := 0 to imageList.Count - 1 do
    begin
            cycImg := imageList.Items[i];
      img := TCyclopsImage.Create(nil);
//        bmps[i] := cycImg.getbitmap;
      img.Picture.Bitmap := cycImg.Picture.Bitmap;//bmps[i];
                        img.copy(cycImg);
      imgList.add(img);
    end;
  mailler:= TfrmMailler.createWith(self, imgList, selectedPatient);
//  viewer.Caption := viewer.Caption + '  ( ' + sgPatient.Cells[0,sgPatient.Row] +' )';
  mailler.ShowModal;


     //release memory

 {   for i := 0 to imageList.Count - 1 do
    begin
      img := imgList.Items[i];
      //bmp := img.Picture.Bitmap;
      img.Parent := nil;
      img.Free;
//        img := nil;
      bmps[i].Free;

    end;
}    mailler.Free;
    imgList.Clear;

end;
//------------------------------------------------------------------------------
procedure TfImageViewer.ToolButton14Click(Sender: TObject);
begin
   imageList.setTool(dtSelectTool);
   showOnSBCurrentTool(dtSelectTool);
end;

procedure TfImageViewer.mnuLInhaClick(Sender: TObject);
begin
    imageList.setTool(dtLineTool);
```

```
    showOnSBCurrentTool(dtLineTool);
end;

procedure TfImageViewer.circulo1Click(Sender: TObject);
begin
    imageList.setTool(dtCircleTool);
    showOnSBCurrentTool(dtCircleTool);
end;

procedure TfImageViewer.retngulo1Click(Sender: TObject);
begin
    imageList.setTool(dtRectangleTool);
    showOnSBCurrentTool(dtRectangleTool);
end;

procedure TfImageViewer.texto1Click(Sender: TObject);
begin
    imageList.setTool(dtTextTool);
    showOnSBCurrentTool(dtTextTool);
end;

procedure TfImageViewer.borracha1Click(Sender: TObject);
begin
            imageList.EraseSelected;
end;

procedure TfImageViewer.ToolButton9Click(Sender: TObject);
var
    imRef    : TCyclopsImage;
    spTmp   : TStatusPanel;
//    pbref   : TPaintBoxEx;
begin
    imageList.setTool(dtWindowTool);
    showOnSBCurrentTool(dtWindowTool);

    spTmp := sbInformations.Panels.Items[7]; //returns the image number

    if spTmp.Text <> '0' then
                //now, we must retreave the image index at imageList
                imRef := imageList.Items[imageList.ImageNumberToIndex(StrToInt(spTmp.Text))]
    else
            imRef := imageList.Items[0];

//    pnlWindow.Visible := true;
    edtWindowCenter.Text := IntToStr(imRef.WindowCenter);
    edtWindowWidth.Text := IntToStr(imRef.WindowWidth);
end;

procedure TfImageViewer.ToolButton7Click(Sender: TObject);
begin
    imageList.setMagnifyWidth(StrToInt(edtMagWidth.text));

    imageList.setTool(dtMagnify);
    showOnSBCurrentTool(dtMagnify);

//    pnlZoom.Visible := true;

end;

procedure TfImageViewer.ToolButton5Click(Sender: TObject);
begin
            Mailer1Click(self);
end;

procedure TfImageViewer.ToolButton4Click(Sender: TObject);
begin
            PrinterTool1Click(self);
end;

procedure TfImageViewer.ToolButton18Click(Sender: TObject);
```

```
begin
   pnlFlip.Visible := true;
end;

procedure TfImageViewer.ToolButton17Click(Sender: TObject);
begin
            if (imageList.Count > 0 ) then
   begin
            sbImages.VertScrollBar.Position := 0;
      sbImages.HorzScrollBar.Position := 0;
      frmLayout.imageList := imageList;
       frmLayout.sbMain := sbImages;
      frmLayout.ShowModal;
      sbImages.VertScrollBar.Increment := frmLayout.imagesHeight;
      sbImages.HorzScrollBar.Increment := frmLayout.imagesWidth;
   end else
    begin

   end;
end;

procedure TfImageViewer.cor1Click(Sender: TObject);
var
            ret                  :           boolean;
begin
   ret := ColorDialog1.Execute;

   if (ret) then
    begin
//            shpColor.Brush.Color := ColorDialog1.Color;
      imageList.setColor(ColorDialog1.Color);
   end;

end;

procedure TfImageViewer.FormShow(Sender: TObject);
var
            imRef : TCyclopsImage;
begin

    imRef := imageList.Items[0];
//    pnlWindow.Visible := true;
   edtWindowCenter.Text := IntToStr(imRef.WindowCenter);
   edtWindowWidth.Text := IntToStr(imRef.WindowWidth);
end;

end.

//----------------------------------------------------------------------------------------------------

unit uMailler;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus, ComCt rls, ExtCtrls, Buttons, uCyclopsImageList,
  uCyclopsImage, ImgList, ToolWin, Jpeg, SakSMTP, SakMsg,sak_util, sakMime,
  sakRegister,sakpop3, UMaillerOptions, Psock, NMsmtp, DIMimeStreams,
  UCyclopsPatient;

type
  TfrmMailler = class(TForm)
    scbThumbs: TScrollBox;
    MainMenu1: TMainMenu;
    mmMessage: TMemo;
    Panel1: TPanel;
    Mail1: TMenuItem;
    NewMail1: TMenuItem;
    Save1: TMenuItem;
```

```
    N1: TMenuItem;
    Quit1: TMenuItem;
    Config1: TMenuItem;
    Config2: TMenuItem;
    ImageList1: TImageList;
    Panel2: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    edtTo: TEdit;
    edtCC: TEdit;
    edtSubject: TEdit;
    lbAtached: TListBox;
    Label4: TLabel;
    btnSend: TBitBtn;
    popAttach: TPopupMenu;
    deleteall1: TMenuItem;
    N2: TMenuItem;
    delete1: TMenuItem;
    btnCancel: TBitBtn;
    Label5: TLabel;
    NMSMTP1: TNMSMTP;
    edtSend: TEdit;
    Timer1: TTimer;
    procedure lbAtachedDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure lbAtachedDragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure mmMessageChange(Sender: TObject);
    procedure delete1Click(Sender: TObject);
    procedure deleteall1Click(Sender: TObject);
    procedure btnSendClick(Sender: TObject);
    procedure Config2Click(Sender: TObject);
    procedure NewMail1Click(Sender: TObject);
    procedure scbThumbsMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure NMSMTP1SendStart(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
                        imageList  : TList; //lista de todas imagens
      attachList        : TList; //lista de imagens anexadas ao mail
      SakSMTP                   : TSakSMTP;
      mail              : TSakMsg;
      cabImagens, imagens : String;
      teste     : TMemo;
                        patient                 : TCyclopsPatient;
      //messages
      procedure smtpError(Sender: TObject; Error: Integer; Msg: string);
      procedure smtpAfterSend(Sender: TObject);
      procedure smtpBeforeSend(Sender: TObject);
      procedure smtpSendProgress(Sender: TObject; Percent: Word);

      procedure imageMouseUP(Sender: TObject;  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

      function  geraHTML(mensagem : string): String;
  public
                        constructor             CreateWith(AOwner : TComponent; list : TList; p : TCyclopsPatient);
      destructor        Destroy;
      procedure         clearAllTmpFiles;
      procedure   showThumbs;
  end;

var
  frmMailler: TfrmMailler;

implementation


{$R *.dfm}
```

```
{ TfrmMailler }
//----------------------------------------------------------------------------
//method:                 CreateWith
//description:  Creates a new instance of TfrmMailler and set a list of images to
//be attached.
constructor TfrmMailler.CreateWith(AOwner: TComponent; list: TList; p : TCyclopsPatient);
var
        i: integer;
        newList: TList;
        cycImage: TCyclopsImage;
begin
        inherited create(AOwner);
   patient := p;
   //creates the list that will contain all images.
        imageList := list;

   //display imagens on tne sbImages
    showThumbs;

   //create mail componnets
   SakSMTP := TSakSMTP.Create(self);
   mail := TSakMsg.Create(self);
        //attaching messages enbeded
    SakSMTP.OnError := smtpError;
   //clear tmp image list
        newList.Free;

   teste       := TMemo.Create(self);
   teste.top    := frmMailler.height + 10;
   teste.left   := frmMailler.width  + 10;
   teste.Parent := frmMailler;

   cabImagens   := '';
   imagens      := '';
end;

//----------------------------------------------------------------------------
//method:                 Destroy
//description:  Clean up the instance object. Clear all tmp files (attached) and
//the image list used by tumbies.
destructor TfrmMailler.Destroy;
begin
        //verificar se esta desalocando corretamente a memoria?????
        imageList.Destroy;

   //clear all tmp files
   clearAllTmpFiles;
end;
//----------------------------------------------------------------------------
//method:                 showThumbs
//description:            display the image list as thumbs in a scrollbox
procedure TfrmMailler.showThumbs;
var
        cycImage  : TCyclopsImage;
   count                : Integer;
   offSet               : Integer;
   vertBack             : Integer;
begin
        offSet := 4;
   vertBack := scbThumbs.VertScrollBar.Size;
        for count := 0 to imageList.Count - 1 do
   begin
        cycImage := imageList.Items[count];
      cycImage.Top := offSet;
      cycImage.Left := 4;
      cycImage.Width := 90;
      cycImage.Height := 90;
      cycImage.Stretch := True;
      offSet := offSet + cycImage.Height + 4;
      cycImage.DragMode := dmAutomatic;
```

```
      cycImage.ShowHint := true;
      //with cycImage.Canvas do
      //begin
                        //              Font.Color := clRed;
      //    Font.Size := 50;
      //    Brush.Color := clBlack;
      //    TextOut(      cycImage.Picture.Bitmap.Width - 70,
      //                                cycImage.Picture.Bitmap.Height - 70,
      //                                        IntToStr(cycImage.imageNumber));
      //end;
      cycImage.Parent := scbThumbs;
   end;

end;
//---------------------------------------------------------------------------
//validate the source and target draging objects
procedure TfrmMailler.lbAtachedDragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
   Accept := Source is TCyclopsImage;
end;
//---------------------------------------------------------------------------
//executes the drag action
procedure TfrmMailler.lbAtachedDragDrop(Sender, Source: TObject; X,
  Y: Integer);
var
            path          : String;                  // where the exe file is locatade
   name     : String;      // the new name used by the attached file
   count    : Integer;      // used to numeric interaction
   jpegIm   : TJPEGImage;
   Aux      : string;
begin
            if (Sender is TListBox) AND (Source is TCyclopsImage) then
   begin
      //extracts the exe path file and creates the new file name
      path := ExtractFilePath(Application.ExeName) + 'tmp';
      name := 'pic_' + IntToStr((Source as TCyclopsImage).imageNumber);

      //verify if exists a duplication
      for count := 0 to lbAtached.Items.Count -1 do
      begin
         if (lbAtached.Items[count] = (name) ) then
            Exit;
      end;
      //save the file as bmp in the temp file
      jpegIm := TJPEGImage.Create;
      jpegIm.Assign((Source as TCyclopsImage).Picture.Bitmap);
      jpegIm.SaveToFile(path + '\' + name + '.jpg');

      MimeEncodeFile(path + '\' + name + '.jpg', path + '\' + name + '.enc');

      cabImagens := cabImagens + ' ';
      cabImagens := cabImagens + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0">';
      cabImagens := cabImagens + '<TR>';
      cabImagens := cabImagens + '<TD align="CENTER" width="33%" valign="MIDDLE" class="tittle" colspan="2">Image
Number: 1</TD>';
      cabImagens := cabImagens + '<TD align="CENTER" width="33%" valign="MIDDLE" class="tittle" colspan="2">Slice
Position: -24.00</TD>';
      cabImagens := cabImagens + '<TD align="CENTER" width="33%" valign="MIDDLE" class="tittle" colspan="2">Slice
Thickness: 5.0</TD>';
      cabImagens := cabImagens + '</TR>';
      cabImagens := cabImagens + '</TABLE>';
      cabImagens := cabImagens + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0">';
      cabImagens := cabImagens + '<TR><TD align="CENTER" width="100%" valign="MIDDLE" class="image">';
      cabImagens := cabImagens + '<IMG BORDER=0 ALIGN=MIDDLE SRC="cid:' + name + '.jpg" ALT="' + name + '.jpg"';
      cabImagens := cabImagens + '</TD>';
      cabImagens := cabImagens + '</TR>';
      cabImagens := cabImagens + '</TABLE>';

      aux := imagens;
```

```
      aux := aux + #13 + #10 + '--320137920059543 '+ #13 + #10;
      aux := aux + 'Content-type: image/jpeg;name=' + name + '.jpg;charset=iso-8859-1 '+ #13 + #10;
      aux := aux + 'Content-id: <' + name + '.jpg> '+ #13 + #10;
      aux := aux + 'Content-disposition: inline;filename=' + name + '.jpg ' + #13 + #10;
      aux := aux + 'Content-transfer-encoding: base64' + #13 + #10;

      teste.lines.LoadFromFile(path + '\' + name + '.enc');
      aux := aux + #13 + #13 + #10 + teste.lines.Text + #13 + #13 + #10;
      imagens := aux;

      //show images atached
      lbAtached.AddItem(name,nil);

     //attach image on the mail
//      mail.AttachedFiles.add(path + '\' + name + '.jpg');

   end;
end;
//---------------------------------------------------------------------------
//clear all tmp files created to compose de mail message
procedure TfrmMailler.clearAllTmpFiles;
begin
            //this method is no more necessary because, in a higher level the tmp
   //directory is all cleaned.
end;
//---------------------------------------------------------------------------
procedure TfrmMailler.mmMessageChange(Sender: TObject);
begin

end;
//---------------------------------------------------------------------------
//method:           delete1Click
//situation:    occurs when the delete popup menu is selected throw the attached
//list box
//description:      clear the selected item at the list box
procedure TfrmMailler.delete1Click(Sender: TObject);
begin
   if (lbAtached.ItemIndex) > -1 then //verify if a item is selected
   begin
      lbAtached.Items.Delete(lbAtached.ItemIndex); //delete de selected index
   end
end;
//---------------------------------------------------------------------------
//method:           deleteall1Click
//situation:    occurs when the delete all popup menu is selected throw the a
//ttached list box
//description:      clear all items at the list box
procedure TfrmMailler.deleteall1Click(Sender: TObject);
begin
   lbAtached.Items.Clear; // clear all elements
end;

//---------------------------------------------------------------------------
// gera inicio HTML
function TfrmMailler.geraHTML(mensagem : String): String;
var
  Aux : string;
begin
  aux :=      '--320137920059543 ' + #10;
  aux := aux + 'Content-type: text/html;charset=iso -8859-1 ' + #10 + #10;
  aux := aux + '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> ' + #10;
  aux := aux + '<HTML> ' + #10;
  aux := aux + '<HEAD> ' + #10;
  aux := aux + '<meta http-equiv="Content -Type" content="text/html; charset=iso -8859-1"> ' + #10;
  aux := aux + '<meta name="Cyclops Project" content="Mail User Agent"> ' + #10;
  aux := aux + '<title>Cyclops Project Mail User Agent</title> ' + #10;
  aux := aux + '<style><!-- ' + #10;
  aux := aux + 'BODY {font -family: Lucida,Verdana,Helvetica,Arial,sans-serif; font -size: 10pt; color: #FFFFFF; background-color:
#000000} ' + #10;
```

```
  aux := aux + 'TD {font-family: Lucida,Verdana,Helvetica,Arial,sans-serif; font-size: 10pt; color: #000000; background-color:
#FFFFFF} ' + #10;
  aux := aux + 'TD.visit {font-family: Lucida,Verdana,Helvetica,Arial,sans-serif; font-size: 10pt; color: #FFFFFF; background-color:
#000000} ' + #10;
  aux := aux + 'TD.tittle {font-family: Lucida,Verdana,Helvetica,Arial,sans-serif; font-size: 10pt; color: #000000; background-color:
#FFCC00; font-weight: bold} ' + #10;
  aux := aux + 'TD.image {font-family: Lucida,Verdana,Helvetica,Arial,sans-serif; font-size: 10pt; color: #FFFFFF; background-color:
#000000; font-weight: bold} ' + #10;
  aux := aux + 'A {text-decoration: underline; font-weight:normal; color: #FFEE22;} ' + #10;
  aux := aux + 'A:visited {text-decoration: underline;font-weight: normal;color: #666666;} ' + #10;
  aux := aux + 'A:active { text-decoration: underline; font-weight: normal;color: #FF0000;} ' + #10;
  aux := aux + 'TD.comments {font-family: Lucida,Verdana,Helvetica,Arial,sans-serif; font-size: 10pt; color: #000000; background-
color: #FFFFFF; font-weight: none} ' + #10;
  aux := aux + 'LI {font-family: Lucida,Verdana,Helvetica,Arial,sans-serif; font-size: 10pt; color: #000000; background-color:
#FFFFFF} ' + #10;
  aux := aux + '--></style> ' + #10;
  aux := aux + '</HEAD> ' + #10;
  aux := aux + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0"> ' + #10;
  aux := aux + '<TR><TD align="CENTER" valign="MIDDLE" class="image"> ' + #10;
  aux := aux + '<IMG BORDER=0 ALIGN=LEFT SRC="cid:cyclops.jpg "ALT="Cyclops Logo"> ' + #10;
  aux := aux + '</TD> ' + #10;
  aux := aux + '<TD align="CENTER" width="70%" valign="MIDDLE" class="image"> ' + #10;
  aux := aux + 'Cyclops DICOM Mail Tool ' + #10;
  aux := aux + '</TD> ' + #10;
  aux := aux + '<TD align="CENTER" valign="MIDDLE" class="image"> ' + #10;
  aux := aux + '<IMG BORDER=0 ALIGN=RIGHT SRC="cid:medicalClinic.jpg" ALT="Medical Clinic Logo"> ' + #10;
  aux := aux + '</TD> ' + #10;
  aux := aux + '</TR> ' + #10;
  aux := aux + '</TABLE> ' + #10;
  aux := aux + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0"> ' + #10;
  aux := aux + '<TR> ' + #10;
  aux := aux + '<TD align="CENTER" width="100%" valign="MIDDLE" class="tittle">Mensagem</TD> ' + #10;
  aux := aux + '</TR> ' + #10;
  aux := aux + '<TR><TD align="LEFT" width="100%" valign="MIDDLE" class="comments"> ' + #10;
  aux := aux + '<PR> ' + #10;

  aux := aux + mensagem + #10;

  aux := aux + '</PR>' + #10;
  aux := aux + '</TD>' + #10;
  aux := aux + '</TR>' + #10;
  aux := aux + '</TABLE>' + #10;

  // anexos cabecalho
  aux := aux + cabImagens + #10;


  aux := aux + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0">' + #10;
  aux := aux + '<TR><TD width="100%" valign="TOP" class="tittle" align="CENTER">Patient' + "" + 's Information</TD></TR>' +
#10;
  aux := aux + '<TR>' + #10;
  aux := aux + '<TD ALIGN=LEFT WIDTH="100%">' + #10;
  aux := aux + '<UL>' + #10;
  aux := aux + '<LI> Patient' + "" + 's Name: '   + patient.data.patientsName + '</LI>' + #10;
  aux := aux + '<LI> Patient' + "" + 's ID: '     + patient.data.patientID + '</LI>' + #10;
  aux := aux + '<LI> Patient' + "" + 's Birth Date: '   + patient.data.patientsBirthDate + '</LI>' + #10;
  aux := aux + '<LI> Patient' + "" + 's Sex: '    + patient.data.patientsSex + '</LI>' + #10;
  aux := aux + '<LI> Patient' + "" + 's Age: '    + " + '</LI>' + #10;
  aux := aux + '<LI> Patient' + "" + 's Size: '   + " + '</LI>' + #10;
  aux := aux + '<LI> Patient' + "" + 's Weight: ' + " + '</LI>' + #10;
  aux := aux + '</UL></TD>' + #10;
  aux := aux + '</TR>' + #10;
  aux := aux + '</TABLE>' + #10;
  aux := aux + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0">' + #10;
  aux := aux + '<TR><TD width="100%" valign="TOP" class="tittle" align="CENTER">Series General Information</TD></TR>' +
#10;
  aux := aux + '<TR>' + #10;
  aux := aux + '<TD ALIGN=LEFT WIDTH="100%">' + #10;
  aux := aux + '<UL>' + #10;
  aux := aux + '<LI> Modality: CT</LI>' + #10;
```

```
  aux := aux + '<LI> Series Instance UID: 1.2.840.113619.2.22.287.1.403.2.20010707.183839</LI>' + #10;
  aux := aux + '<LI> Series Number: 2</LI>' + #10;
  aux := aux + '<LI> Series Date: July 7, 2001</LI>' + #10;
  aux := aux + '<LI> Series Time: 8:38:39 am</LI>' + #10;
  aux := aux + '<LI> Protocol Name: Unknown</LI>' + #10;
  aux := aux + '<LI> Body Part Examined: HEAD</LI>' + #10;
  aux := aux + '<LI> Operators Name: #(' + '"' + 'MRC ' + '"' + ' nil nil nil nil)</LI>' + #10;
  aux := aux + '</UL></TD>' + #10;
  aux := aux + '</TR>' + #10;
  aux := aux + '</TABLE>' + #10;
  aux := aux + '<TABLE width="100%" cellpadding="5" cellspacing="0" border="0">' + #10;
  aux := aux + '<TR><TD align="CENTER" width="100%" valign="TOP" class="visit">' + #10;
  aux := aux + '<PR>' + #10;
  aux := aux + 'Visit the Cyclops Project Web Page at: <A
href="http://www.inf.ufsc.br/cyclops">http://www.inf.ufsc.br/cyclops</A></PR>' + #10;
  aux := aux + '</TD>' + #10;
  aux := aux + '</TR>' + #10;
  aux := aux + '</TABLE>' + #10;
  aux := aux + '</BODY>' + #10;
  aux := aux + '</HTML>' + #10;

// anexos imagens
  aux := aux + imagens + #10;

// anexos logos
  aux := aux + #10 + '--320137920059543 ' + #10;
  aux := aux + 'Content-type: image/jpeg;name=medicalClinic.jpg;charset=iso -8859-1 ' + #10;
  aux := aux + 'Content-id: <medicalClinic.jpg> ' + #10;
  aux := aux + 'Content-disposition: inline;filename=medicalClinic.jpg ' + #10;
  aux := aux + 'Content-transfer-encoding: base64 ' + #10 + #10;

//logo DMI
  aux := aux + '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a ' +
#10;
  aux := aux + 'HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy ' + #10;
  aux := aux + 'MjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjL/wAARCABLAQsDASIA ' + #10;
  aux := aux +
'AhEBAxEB/8QAHAAAAQUBAQEAAAAAAAAAAAAAAAMEBQYHAQgC/8QAORAAAQMDAgQDCAIBAQkBAAAA ' +
#10;
  aux := aux + 'AQIDBAAFEQYSEyExUUGRoQcUIjJSYXGBFTNCIxYlU2JygrHB0eH/xAAaAQEAAwEBAQAAAAAAAAAA '
+ #10;
  aux := aux +
'AAAAAgMEAQUG/8QAJREBAAICAgIBAwUAAAAAAAAAAAECAxESIQQxUQUTQRQyYbHB/9oADAMBAAIR ' + #10;
  aux := aux + 'AxEAPwDBFKVvPxHr3r53K+o+dCvnP5rlccd3K+o+dG5X1HzrlFB3cr6j50blfUfOuUUHdyvqPnRu ' + #10;
  aux := aux + 'V9R865RQd3K+o+dG5X1HzrlFB3cr6j50blfUfOuUUHdyvqPnRuV9R86+mmlvOoabQVuLUEpSkZJJ ' + #10;
  aux := aux + '6AVfZHsiv8SAt1+Vbm5iGC+qAZI4wQBn5fwKCgblfUfOjcr6j51fF+zj3Q6ZenXVtuJeo/vJWlpS ' + #10;
  aux := aux + 'lMp2hWMDqeeKqlythZuD7cFuW/FTlTTrjBSVo+rHagjtyvqPnRuV9R86mYljYk6ZmXY3Jtp6O4EJ ' + #10;
  aux := aux + 'iqbVucB8QrpUY7DlMhBdjPNhZwje2Ru/HegR3K+o+dG5X1HzpV+JJi494jvM7unEQU586s2kdBT9 ' + #10;
  aux := aux + 'YQp8yLMhxWIJTxlyXNgG7ODn9UFU3K+o+dG5X1Hzq33H2cXu2362Wxwx1ouSgmLLac3MuZ7KFOz7 ' + #10;
  aux := aux + 'J9RjW3+y+xr3rgh8PZPDKMdc/nlQUXcr6j50blfUfOtAtnsku9zkXeOi421p21PqZkpdewRtx8X/ ' + #10;
  aux := aux + 'AE8+v2qhPtFiQ4ypSVFCiklJyDg+BoPjer6j50b1fUfOvk1L6b03cdUXVEC3NblnmtZ+VtPcmu6O ' + #10;
  aux := aux + 'oRW9Xc1P2zRmprw2HYVplraV0cUnYk/s4rb9O6CsOk2kLTHRPuI+aS+nISf+RJ5D89anX5Mh4/E4 ' + #10;
  aux := aux + 'r8A8qtrhmfbDl82tOqxthKvZZrBKSTBRy8BIT/8AagLnpu+2Ybp9ukso+spJT5jlXoSQ4WGy448U ' + #10;
  aux := aux + 'pHUk1AzZ96uLao1ngOuhQxxpJ2NeR5mpzhiPyz0+oZLTrj/jA96u586N5+o+daW97JZykuyJV2hI ' + #10;
  aux := aux + 'fVlXCbScZ7fas8uNuk2uc5ElNlDqDzHf7j7VTNJjuXo482PJPGs7k33q+o+dOGlK4Y+I+dNRThv+ ' + #10;
  aux := aux + 'sVBaRV85/NcqRs1mlX++R7XD2e8SVlKN5wnOCeZ/VSkzRNwiFlaZUCRFcdDSpMd8LbaUem/HNPlQ ' + #10;
  aux := aux + 'VqilXI7rZVlJKUkjcnmnzr54TmwL2K2noccqD4or6U2tAypCgPuMVJz7ZHiQLetqUXpclsrdYSj+ ' + #10;
  aux := aux + 'rmQBnvgZoIqipSNp+fJiMTA2G4j0j3YPOHCUr6nd2Az1qPcYW2txONwbUUlSeY86BOivooUkAlJA ' + #10;
  aux := aux + 'IyCR1qWRpq4uaWXqJCUKgof4CiFfEFYBzjtzHOgh6KnbzpS42C5QYNwDaHZjKHk7V5CUqOPi7EYO ' + #10;
  aux := aux + 'aP4BuXq9Njtkv3hC5IjokqQQCScbiB4ZoEdLXNmzaptlxkIK2Y8hDiwBnkDzrSdZ6Md1DqS56rg6 ' + #10;
  aux := aux + 'htotEhBkNPGT8YO35NvUHIx+6zC4Wh63lIU606pTjjRQ2cqSUKwcjwz4UxCXMKwFYT832/NB6WsE ' + #10;
  aux := aux + '6Gmd7NyuZHw3Z1ocJdHwq4aOR58jT6VqYxtX6NtzNxjC3SYaky0ZQUkhIwFE9PxmvLY3n5dxx2ru ' + #10;
  aux := aux + '13eEFK9/04OaD0Bo+XYLdZNTIuRii3t6gJLJwQGwrkQn6Ryo1G5d39aw3rvqK2o00u5IXBLKm1La ' + #10;
  aux := aux + 'GPhIGOXgDmvP21wg8lffAqcsWlpmooU6THlRGkQkcR1L72w7e4/8fug272tPx5ns1mJfciuyY9wT ' + #10;
  aux := aux + 'wCJCXnOHywrI+XIzyqoeyiN/I6B1pa25EduVLQ0hpLzwQFH4u9ZMW3umxZ546HnXA28AohDmE8jg ' + #10;
  aux := aux + 'HlQbszIg6at+idJybhFk3KPcfeH1tOBSGASTt3fvFWBXtOio9oqrGttr3tNw93TPyNvupwvaVfnl ' + #10;
  aux := aux + 'XmjY6pJWErIHVWCRXdrhTxNqsHorB50HoewwDK1L7SZCJcNKJy3YzG+QlO5fXP45jnXnh9lcd9xl ' + #10;
  aux := aux + 'zG9tRSrBzzFdDb+fkcySfA86SPU560AlJWtKQQCTjJOBXonScjTGj9OMQGbnEVMeAXJcbXuW6s+A ' + #10;
  aux := aux + 'A54HQCsq9m+h29a3p5mVIWxDjIDjqkD4jzwAK9CWfSVh0+0EWi2MMqxgvqTucV+VHnVlPlRnmJjW ' + #10;
  aux := aux + 'zJiWmUN7cd9KMcluI2Z/R51xziK+VIAqwKhpJyoE0g4wE5wAKvizzL4plXlRldVdfvSbgWBjccds ' + #10;
```

```
  aux := aux + '1JyFtIzuUM/bnURLW86NsZSWyeq1pzj8CrYljvSsdGEooabU44tKEJ5lSjgD91luuLra7qttmEDI ' + #10;
  aux := aux + 'ltnHFbHLb2+9aY5p63PrDt0VIuSxzCH14bH4QMCnBktQmCxAhRYjeMYaaCf/AAK7aLXjjCGO2Hx7 ' + #10;
  aux := aux + 'xmmZmY+OnnLBBIPXtS7f9YqV1dBEDUUlKRhDuHU/g/8A7mopv+sVgtXjMxL6XFkjLjJX1MbWX2c ' + #10;
  aux := aux + 'SGYvtHs70h5tlpL6tzjitqRlKhzNXPQ0ZrQt63Xi4QSq4PssoYZkJcCU8QKK1EcgBjxrIVfOfzRk ' + #10;
  aux := aux + '1FY3fTv8Yiwux5N0TKt77MkvR3ZqEtMuHfhsNYyTnBByOfSvi6PSHbTfnmp8V7TgXFVbWmXEkMI4 ' + #10;
  aux := aux + 'ieqRzScdc8+tYXk1PXDWl+ulnRapU0GEgJHDbaQ3uwMDcUgbv3QanK1Fbp99uDl4mxX48S+lMPft ' + #10;
  aux := aux + 'KWmyhYSoAdU7tpPX704ss6Y1ELV8nwZl9dWrgymJrSH2GCP8HCMdf8T4VguTXdxoPQVpusgzERLd ' + #10;
  aux := aux + 'qhl5uNekuS1vvNtFxgoTuyOQVzBBI6kZ8arL0xS7bF/j58FFjYUv+YhlxAW8eMoq+E81ZTjBHTwx ' + #10;
  aux := aux + 'WR5NczQazq2Yt6xX5dzukCZBedSbI0wtClNJ38sADKRs5EGmWl9TwtP6Eh++NImMLuLwkw9+CpBQ ' + #10;
  aux := aux + 'kpJGegUB61meaM0G8WzUMCWIshx9py8P2lkNr96SyoHiLK0hwghKuY7VJLuK1Xa0P2yfbLW+iUhd ' + #10;
  aux := aux + '6HvjalvIGAlSlgAKJAOceNedcmu5Peg29m4x1aZEe1XCFHv6npYivOOJSUAv5ICjySSMEfakbdNn ' + #10;
  aux := aux + 'f7uQ3ebW3wJTitQ7nGwJJ3g7sY+MFHLl4gn71i2TUrpyxP6jvTVuZdQ0VhS1uL6JQkZUeXXkOlBb ' + #10;
  aux := aux + 'fZ5sV7Q7kbWUIT7tLMUuEbU8jsyTy5cutW56e8m1NRP5W3jXBigGfxW8FHEJCeJ8u4Jxz645Zqi2 ' + #10;
  aux := aux + 'u0tQdQNt2K+OzuOw6097nFJdCSMKTsV4EHr9qi5+k5qLwq22xibOkNpKnUiMpKkczjI8OWKDQNX6 ' + #10;
  aux := aux + 'hRA0Zw7dcYxuzjzTVzcjlB4rnBwsjA6HoSOpzVX0FCVJsmqVB+K2XIBaQl15KFKVuCuQPXkDVKUy ' + #10;
  aux := aux + 'tqRwZAUypKtqwpJBTz55FWW56JmMPW9Nlccu7c2N7y2qOyoEJCik5HhzFBfZmrn40O5MRLkylEWx ' + #10;
  aux := aux + 'w1w0p2fC/tQFlPdY5jPUVZYcm2Kvapqb026zLUwqdGE1tllOWWwpakkHiFRzkD91gsbT15miQY1r ' + #10;
  aux := aux + 'lu+7cntrR/0/Hn2qQOnlxdJPXSdBuLby3UpjL4ZS1s55KiR3wBQasFFbcxFmmxDphuzupVDadTuD ' + #10;
  aux := aux + '4Sd5Unrnd4n7Y8K+7pebfIsTwgsxV2ZTLaWEruDe1rG3mlnG5Kgc+PfvWRI1pf2rB/CNzgiBs2FC ' + #10;
  aux := aux + 'GkJUU5zgrA3Efk0vYdLM3Kyybvcbq3boDLwYC1NqWVOEZxgfag0m4azkNzr0pq6sgMXqO3FKSj4I ' + #10;
  aux := aux + '6s79h7Hlkjr41luszHOtr0YpbMczHS2WsbSnccYxyxUjE0YwYLtyuF8jRbZ7wqPHkcNS/eFDmSlI ' + #10;
  aux := aux + '54+5qu3SCm23J6KiUzKQjG19k5QsEAgjzoNU9i18s1khXdVynsRXHVthPFOMpAPT91qLOv7LK3N2 ' + #10;
  aux := aux + 'lEq6vDltiskpz91HAFYn7INOWPUeoZLd5aL/AAGg60wVYSs5wd2Oo+1ek4Vugx2EsMMNsMJGEtMp ' + #10;
  aux := aux + 'CEjyqyJnSi8Ry/lCG63R8JJhMwweqXFcRfpypN1ubK+Yq2/bkKtSmYrQ/wBNpIPfGTUfLcQjJUoD ' + #10;
  aux := aux + '8mrKyy5ce/3Srqrfs+c8+wpu40hHRNPpM1vJCMqP2qLedcc6fCPtV0befk4R1CPuEyLCRvkvoaT4 ' + #10;
  aux := aux + 'bjzP4HjValXG73LKLHZpD+eQffTw0evM1ZvdY7Dpe4Danj1cUnKvM0lKmPlJSHFJHYcqnEWn1OmS ' + #10;
  aux := aux + '18VO713/AExDVtvvEG6JN5WhUh1OQEKyEgHpUM3/AFirN7QZIe1AloHPBZAJ+5JNVlv+sViyxq8w ' + #10;
  aux := aux + '+j8O83wVtMa3Hoir5z+a5XVfOfzXKraRRRRQFFFFAUUUUBRRRQFFFFAVLaacbZvjDi7ou2KRlTct ' + #10;
  aux := aux + 'CSrhr8M48O9RNFBsKbhpN2VHXerpAm3xAWoTYza2GFk42h4thJUc5JOKd6i1HY7xbmbRb79CtskM ' + #10;
  aux := aux + 'tFcxh LiWlFBXlvPNWBkEZzWJ0UFr1RJt+o9crXHmNsRnQ20qY8CEqUlASXFY58yM/urlJvNlc05G ' + #10;
  aux := aux + '0nC1EzFkRmGgboErS05hS1KQCBux8Y69qyKig29rVmm5ipCJ1yjuQo5QdwU6xKccS0lKnUKTyVuK ' + #10;
  aux := aux + 'eiqQ1PrK2XOz3d+3zLKlqaxsDKkPe9Y5YSQTsBGOoFYvRQTFovUa2RZLD1kt88vD4XZKV72+WPhK ' + #10;
  aux := aux + 'VD81O6Kme6x5IZ1LFtjjiwHos2PxWH28cuRBBOfDH7qlUUGn3yVpTU8UWi33Jm0tQJTjsdT6FBp1 ' + #10;
  aux := aux + 'DgSV4wCU/GDgdjWfXeNDiXR5i3yzLioICHynbv5DJx4c80xooJzSOo3dK6jj3RtBcCMhxsHG9J6i ' + #10;
  aux := aux + 'tzsftHv+qUn+B0sstg4MiS9tbT+wOdebz1rW/Zf7SBbI6NP3V7ZFBPu7x5BBP+J+33qdfhXkr1y1 ' + #10;
  aux := aux + 'trrbV+dRm53NsFQILEJvakf95yryxXEwENN7E5CRz+JRUT+SeZr7eu0NlviOy2EIIzuU4MVFDVFv ' + #10;
  aux := aux + 'lulqC6qa4DgiMgrA/J6CrojTzrzNz9bTaPvTN5QBOMAUsVPONgqaLaj1STkimryD/mvH2q2umLJ1 ' + #10;
  aux := aux + '6RU+4tx1bNjrzp6NtIKiapupZmqUwXZDEJuBGSMlx1wFzH46Cru/JaitKc3BITzKlHAFY/rfWCry ' + #10;
  aux := aux + '8YURw+6IPxK/4h/+UyW417lzxcf3ckRFd/Mz6hT3HnH3VOurK1qOSpRyTSjf9YpuKcN/1isMvo/R ' + #10;
  aux := aux + 'FXzn81ynKmkb1cvHvXzw0dvWgQopfho7etHDR29aBCil+Gjt60cNHb1oEKKX4aO3rRw0dvWgQopf ' + #10;
  aux := aux + 'ho7etHDR29aBCil+Gjt60cNHb1oEKKX4aO3rRw0dvWgQopfho7etHDR29aBCil+Gjt60cNHb1oEK ' + #10;
  aux := aux + 'KX4aO3rRw0dvWgQopfho7etHDR29aBCil+Gjt60cNHb1oGs5rnOnPDR29aOGjt60NlIExtqew5OaX ' + #10;
  aux := aux + 'KjIUCtguFO8ds+FbZafa5ppuC3FagfxiEjAaS3lI/Y/91h/DR29aOGjt61KLaV5McXjUt6e9pWn1 ' + #10;
  aux := aux + 'jP8AJpA7BBH/AKqvXP2o2lpKvdGnpLnhy2p8zWTcNHb1o4aO3rVn3rfhl/QYpndpmUvfdXXS/KKX ' + #10;
  aux := aux + '3A1H8GW+Sf33qApxw0dvWjho7etVzaZ7lrpSuOONY1BuKcN/1ijho7etOG2kbBy9aim//9k= ' + #10;
  aux := aux + #10 + '--320137920059543 ' + #10;
  aux := aux + 'Content-type: image/jpeg;name=cyclops.jpg;charset=iso-8859-1 ' + #10;
  aux := aux + 'Content-id: <cyclops.jpg> ' + #10;
  aux := aux + 'Content-disposition: inline;filename=cyclops.jpg ' + #10;
  aux := aux + 'Content-transfer-encoding: base64 ' + #10 + #10;

//logo cyclops
  aux := aux + '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0a ' +
#10;
  aux := aux + 'HBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIy ' + #10;
  aux := aux + 'MjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjL/wAARCABLAHgDASIA ' + #10;
  aux := aux +
'AhEBAxEB/8QAHAAAAgIDAQEAAAAAAAAAAAAAAAAAAUEBgEDCAcC/8QAPRAAAgEDAQUFBAgCCwAAAAAA ' + #10;
  aux := aux + 'AQIDAAQRBQYSITFhEyJBUXEHFUKBFDJDUnKRsdFUoRckMzREVXOCk8Hh/8QAGgEAAgMBAQAAAAAA
' + #10;
  aux := aux +
'AAAAAAAAAwQAAgUGAf/EADARAAEDAgQEAwcFAAAAAAAAAAEAAgMEEQUSITETQVHwIjKBBhQjYXGR ' + #10;
  aux := aux + 'oRUzscHR/9oADAMBAAIRAxEAPwDn+iitkEEtxKI4Y2dzyVRk1F6ASbBa8VndlI8Kt+l7LxQoJdTXe ' + #10;
  aux := aux + 'k+GFW5fixT+O2jICw2kSRr4JGB+dIyVzGGwF1tU+BzyszvOX+V5mkMkjbscbO3kozW33fefwk/8A ' + #10;
  aux := aux + 'xmvSGmjtiUa5tLdvumRUNaG1SwiOX1SDePDI kLfpQ/f3nys7+yY/Q4W+eYA+n+rzp4Jom3ZI2RvJ ' + #10;
  aux := aux + 'hg18FSOYr0b3xpn+ZQHqVJ/UUS2FlqCi4khjuEI7siHw+VWFcR52EIZwRr/2ZQ499CV5zg1irbe7 ' + #10;
  aux := aux + 'LJKxezlWPP2b8vkaS32g31hlnQPGPjjO8P3pplRG/Y6rNnw6pgvnabDnySyiskYrFGSSKKKKiiKK ' + #10;
  aux := aux + 'KKii+4YnnmSKNSzucKB4mvQdJ0pNKgEUI3rqQDtZPLoOlU/Z+eG31mCSZgq8QCeQJGBTnX9fjEb2 ' + #10;
  aux := aux + 'Ni4ZXGJZx8XQdKRqmySOETdua3MLfT00bqmU3cNAOffzU/VNfttMBjh3Lq68TnKJ6+ZpAtzrW0U5 ' + #10;
  aux := aux + 'hR3kUZYouFRB16etIzxNX3R7cW+l21tEFElyA8jDm2TwHyFEZBHCNBc9Vk4tjVQ5t72B2HL16qLB ' + #10;
```

```
  aux := aux + 'sdalXNxqLNuDvtGoCr/uJqFb6Zs8ddtbWTUpGtGYiWXG6q8DjvY88A8K1bS6w1xcvp9uwWzgcqAv ' + #10;
  aux := aux + '2jDgWPn0qunnRg1xGpWZSsnNpJXb8tF0PB7JNidSiaSzuJWEqdzsLwSKnUc8/PNULaL2abQbG2tz ' + #10;
  aux := aux + 'q1nfRT2ULDLRFhIEzzZSMYHDPE86oekareaNqMN7YTtBcRNlXU4+XofKutLC4ttotnYZnCyW97bj ' + #10;
  aux := aux + 'fUHIIYYI/WuWrpqzCnte6TiRuNiCNuwtiLK/y6EdFzGu2FyFAaztnbxYgjP5Gm1ptBptzFvSSfRX ' + #10;
  aux := aux + '5Mrgsp9CKq+0ekyaFtFf6ZIwZraYpvDkw8D+WKV10PusEjQ5gtfomocYq4jq7MOhV8uNE0u/iE8a ' + #10;
  aux := aux + 'pusOElu3d+Yqs6joNzYb0gHawD7RfD1FRNP1K502YyW0m6TwYEZDeoqzWm1lnIgF7A0b8i0Qyp9Q ' + #10;
  aux := aux + 'aplnhPh8Q/KZ4tBWNtIOG/qNu+7qnUVcNR0S21CA3enSIzHJO6eDfLwNVOaGSGQxyIUdTgq3Aimo ' + #10;
  aux := aux + 'pWyDTdZdTSSU5GbUHYjYrXRRRRUqvqNHkkCopLHkBzrMispwwIPkan6MFF5vE4cKd0ZqVqFmbjMi ' + #10;
  aux := aux + '/wBovMedaMOHulpjMw3IO3yQXTBr8pSOvRNIGPdoyT3ErzxgVYgjBHhV20a7MljbTKRvRd38qypR ' + #10;
  aux := aux + 'olMSaTGLKn3n97m/1G/WtFM9asWtLxnGTFId5W9fClIXBuE9G4OYCFkc66f9lAl/o60ppGbiJAoO ' + #10;
  aux := aux + 'eXaNxrnXQdndT2jv1tNMtWmkJ7xH1UHmx5AV1JYR22yOx9ut7LDHFp9mgmdThSyoA2PPJB6nNcx7 ' + #10;
  aux := aux + 'Tygwsp26uJ2596pymHiLjsufPa4iJ7TNVWMALiHl59imapFMNd1W41zW7vU7pszXMhkPkM8gOgHA ' + #10;
  aux := aux + 'elL66GliMUDI3bgAfYJdxuSUUUUUdeKdp2qXGm3AkhbgfrIT3WHUVaZYNN2gRZFlVZMcQMb69D0q ' + #10;
  aux := aux + 'nQ2005xFGznoK32dzLpl8k2530PFG5HpQpqZxHEboeqepK0R/Bl8TDuOnzCzqentp968DNvAYKtj ' + #10;
  aux := aux + 'mCKK+9W1STVLoTSIqYG6qqOQoq8YdlGbdLzcPiO4flvol6ndYEcCKf2N5HdoqM2LgD4vj9OtV+sg ' + #10;
  aux := aux + 'kcqeo619K/M3UcwlZYhILFWC7skufrDckHxY/Wten3V/okzILdZoZProRvA9QRyNR7XWGRQlynaq ' + #10;
  aux := aux + 'OG9nvD96ZwXME/CC5wfuv3TWw+KhxDxNdlceXf8ASTcJGNyPF2pkl/pGqh4HItQOaXTY3vQ8hSiw ' + #10;
  aux := aux + 't9Dj2yt4L2cnSO1UTSISe7jjg4zjP8qkywdquJIN9fMfvUI6Vas3DtU88n/ykpvZ+doIjdcH0KrT ' + #10;
  aux := aux + 'OjicSCbdN11FBHoezemDsPoWn2ON7K7qK3XPxevGufPaT7QH2vv0t7QPFpduT2aNzkbkXb/ry+dI ' + #10;
  aux := aux + '7mwM5RZb65YIMKJCWCjyHHhUf3LEf8Vx/BXP4f7GVFHKZ3+N/UkafndaL66N4sNAktZwcZxTcaGW ' + #10;
  aux := aux + 'c/1hVj8GK8T8qlW+i26AtIzy4P4V+dbkeEVbzbLb6oLqmNvNJ7SxnuyRGndHNicAU2t9Ht0K77PM ' + #10;
  aux := aux + '45hOC1Jlu7K2TcMqBV+yh45pRdavcSsViJiiHJUOPzp0RUNC0GX4j+g27+v2Qs00p8OgTe5v4LLC ' + #10;
  aux := aux + 'ucsPsosAD1qu3dwbq4aVubHl5dK0liedYrOrcQkqtDo0bBGihbHrzRRRRSCMiiiioois5NYoqKLe ' + #10;
  aux := aux + 'l5cxKFSeRVHIBuVSl1q9UAGQNj7yil1FGZUzM8ryPVUMbTuE2TW5VA3oYmPiSOdfXvxv4WH+f70n ' + #10;
  aux := aux + 'opgYnVt0DyqGCM8k1k1q5K4jWKMnxVeNQ5dQu5k3ZJ3ZfImo1FClq55fO8n1V2xMbsEZNFFFLK6K ' + #10;
  aux := aux + 'KKKiiKKKKii//9k= ' + #10;
  aux := aux +  #10 + '--320137920059543-- ' + #10;

  geraHTML := aux;
end;

//---------------------------------------------------------------------------
procedure TfrmMailler.btnSendClick(Sender: TObject);
var
  Aux : string;
begin
  if not(frmMaillerConfig.itsAlreadyRead) then
    begin
      frmMaillerConfig.ShowModal;
    end;

  if NMSMTP1.Connected then
     NMSMTP1.Disconnect;

  NMSMTP1.Host   := frmMaillerConfig.hostname;
  NMSMTP1.UserId := frmMaillerConfig.username;

  NMSMTP1.Connect;

  NMSMTP1.EncodeType := uuMime;
  NMSMTP1.SubType := mtHtml;

  NMSMTP1.PostMessage.FromAddress    := frmMaillerConfig.edtMailUser.Text;
  NMSMTP1.PostMessage.ToAddress.Text := edtTo.Text;

  NMSMTP1.PostMessage.Body.Text      := geraHtml(mmMessage.Text);
  NMSMTP1.PostMessage.Subject        := edtSubject.Text;
  edtSend.Text := 'Enviando e-mail, aguarde...';
  NMSMTP1.SendMail; //Envio do e-mail;
  edtSend.Text := 'e-mail enviado';
  Timer1.Enabled := true;
  NMSMTP1.Disconnect;
end;
//---------------------------------------------------------------------------
//it´s executed when a smtp error ocours
procedure TfrmMailler.smtpError(Sender: TObject; Error: Integer;
  Msg: string);
begin
            showMessage( 'Error: ' + msg);
end;
```

```
//---------------------------------------------------------------------------
procedure TfrmMailler.smtpAfterSend(Sender: TObject);
begin
        showMessage( SakSMTP.replyString);
end;
//---------------------------------------------------------------------------
procedure TfrmMailler.smtpBeforeSend(Sender: TObject);
begin

end;
//---------------------------------------------------------------------------
procedure TfrmMailler.smtpSendProgress(Sender: TObject; Percent: Word);
begin

end;
//---------------------------------------------------------------------------
procedure TfrmMailler.Config2Click(Sender: TObject);
begin
   frmMaillerConfig.ShowModal;
end;
//---------------------------------------------------------------------------
procedure TfrmMailler.NewMail1Click(Sender: TObject);
begin
   mmMessage.Text := '';
   edtTo.Text := '';
   edtCC.Text := '';
   edtSubject.Text := '';
   lbAtached.Clear;
end;

procedure TfrmMailler.scbThumbsMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin

end;
//---------------------------------------------------------------------------
procedure TfrmMailler.imageMouseUP(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
        (sender as TCyclopsImage).Hint := IntToStr((sender as TCyclopsImage).imageNumber);
end;

procedure TfrmMailler.NMSMTP1SendStart(Sender: TObject);
begin
  NMSMTP1.FinalHeader.Clear;
  NMSMTP1.FinalHeader.add('Content-Type: multipart/related;boundary=320137920059543;charset=iso-8859-1');
  NMSMTP1.FinalHeader.Add('This message is in Mime format');
end;

procedure TfrmMailler.Timer1Timer(Sender: TObject);
begin
   Timer1.Enabled := false;
   edtSend.Text := '';
end;

end.

//---------------------------------------------------------------------------------------------------------

unit UMaillerOptions;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TfrmMaillerConfig = class(TForm)
    Label1: TLabel;
```

```
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    edtHostName: TEdit;
    edtUserName: TEdit;
    edtMailUser: TEdit;
    edtPassword: TEdit;
    btnApply: TButton;
    btnOk: TButton;
    btnClose: TButton;
    procedure FormShow(Sender: TObject);
    procedure btnApplyClick(Sender: TObject);
    procedure edtHostNameChange(Sender: TObject);
    procedure edtUserNameChange(Sender: TObject);
    procedure edtMailUserChange(Sender: TObject);
    procedure edtPasswordChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure btnOkClick(Sender: TObject);
  private
          //atributes
    anyChange       : boolean;
    fd                      : TStringList;
    pHostName       : string;
    pUserName       : string;
    pMailUser       : string;
    pPassword       : string;

          //methods
    procedure       loadOldConfiguration;
    procedure       saveCurrentConfiguration;
          function  existsConfFile: Boolean;

  public
    function itsAlreadyRead: Boolean;

  published
          property  hostname  : string    read pHostName;
          property  username  : string    read pUserName;
          property  mailuser  : string    read pMailUser;
          property  password  : string    read pPassword;

  end;

var
  frmMaillerConfig: TfrmMaillerConfig;

implementation

{$R *.dfm}

{ TForm1 }
//---------------------------------------------------------------------------
function TfrmMaillerConfig.itsAlreadyRead: Boolean;
begin
          result := (pHostName <> '') OR (pUserName <> '') OR (pMailUser <> '') OR (pPassword <> '');

end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.loadOldConfiguration;
begin
          if existsConfFile then
    begin
      fd.LoadFromFile(ExtractFilePath(Application.ExeName) + 'mailler.conf');
      edtHostName.Text := fd.Strings[0];
      edtUserName.Text := fd.Strings[1];
      edtMailUser.Text := fd.Strings[2];
      edtPassword.Text := fd.Strings[3];
          end;
end;
```

```
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.saveCurrentConfiguration;
begin
   if existsConfFile then
      DeleteFile(ExtractFilePath(Application.ExeName) + 'mailler.conf');
   fd.Clear;
            fd.Add(edtHostName.Text);
            fd.Add(edtUserName.Text);
            fd.Add(edtMailUser.Text);
            fd.Add(edtPassword.Text);
   fd.SaveToFile(ExtractFilePath(Application.ExeName) + 'mailler.conf');
end;
//---------------------------------------------------------------------------
function TfrmMaillerConfig.existsConfFile: Boolean;
begin
         result := FileExists(ExtractFilePath(Application.ExeName) + 'mailler.conf');
end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.FormShow(Sender: TObject);
begin

   anyChange := false;
   //create the .conf file manipulator
   fd := TStringList.Create;
   //verify if the conf file was already read
   if not itsAlreadyRead then
      loadOldConfiguration;
end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.btnApplyClick(Sender: TObject);
begin
   if anyChange then
   begin
      saveCurrentConfiguration;
      pHostName := edtHostName.Text;
                  pUserName := edtUserName.Text;
      pMailUser := edtMailUser.Text;
      pPassword := edtPassword.Text;

   end;
   Self.Close;
end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.edtHostNameChange(Sender: TObject);
begin
   anyChange := true;

end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.edtUserNameChange(Sender: TObject);
begin
 anyChange := true;
end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.edtMailUserChange(Sender: TObject);
begin
 anyChange := true;

end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.edtPasswordChange(Sender: TObject);
begin
 anyChange := true;
end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.FormCreate(Sender: TObject);
begin
   pHostName      := '';
   pUserName      := '';
   pMailUser      := '';
```

```
   pPassword            := '';
end;
//---------------------------------------------------------------------------
procedure TfrmMaillerConfig.btnCloseClick(Sender: TObject);
begin
    Self.Close;
end;

procedure TfrmMaillerConfig.btnOkClick(Sender: TObject);
begin
    if anyChange then
    begin
       saveCurrentConfiguration;
       pHostName := edtHostName.Text;
                        pUserName := edtUserName.Text;
       pMailUser := edtMailUser.Text;
       pPassword := edtPassword.Text;

    end;
end;

end.

//-----------------------------------------------------------------------------------------------------------

unit UMiniView;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus,UPaintBoxEx, ExtCtrls, UCyclopsImage, ComCtrls, ToolWin;

type
  TMiniView = class(TForm)
    MainMenu1: TMainMenu;
    PopupMenu1: TPopupMenu;
    sbMiniView: TStatusBar;
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private

  public
       im_miniView : TCyclopsImage;
  end;

var
  MiniView: TMiniView;

implementation

{$R *.dfm}

procedure TMiniView.FormCreate(Sender: TObject);
begin

       im_miniView := TCyclopsImage.Create(self);
       im_miniView.Parent := self;

       im_miniView.Top := 0;
       im_miniView.Left := 0;
       im_miniView.Width := Self.ClientWidth;
       im_miniView.Height := Self.ClientHeight;
       im_miniView.Stretch := true;
end;
//---------------------------------------------------------------------------
procedure TMiniView.FormResize(Sender: TObject);
begin
       im_miniView.Width := Self.Width;
```

```
      im_miniView.Height := self.Height;

      sbMiniView.Panels[3].Text := IntToStr(self.Width);
      sbMiniView.Panels[5].Text := IntToStr(self.Height);

end;
//------------------------------------------------------------------------------
procedure TMiniView.FormShow(Sender: TObject);
begin
      sbMiniView.Panels[1].Text := IntToStr(im_miniView.imageNumber);
      sbMiniView.Panels[3].Text := IntToStr(self.Width);
      sbMiniView.Panels[5].Text := IntToStr(self.Height);
end;
//------------------------------------------------------------------------------

end.

//-----------------------------------------------------------------------------------------------------

unit UPaintBoxEx;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  VectorGraphicsListLibrary, VectorGraphicsNodeLibrary, ExtCtrls,LineLibrary,
  stdctrls,clipbrd,ImageLibrary, UCyclopsImage, comCtrls;

type
  TPilotOrient = (tSagital, tCoronal);
  TDrawingState = (dsNotDrawing, dsNewFigure, dsStretchCorner, dsTranslate);
  TMagnify = (mag100,mag200,mag400);
  TPaintBoxEx = class(TPaintBox)

  private
    contResize                                  : Integer;
    comBorda                                    : Bool;
    corDaBorda                                  : TColor;
    espessuraDaBorda                : integer;

    fShowImageNumber                        : boolean;

    imagemDeFundo                           : TCyclopsImage;
    ferramenta                                  : TDrawingTool;
    estadoDoDesenho                         : TDrawingState;
    listaDeFiguras                          : TVectorGraphicsList;
    figuraCorrente                          : TVectorGraphicsNode;      //representa a figura que esta sendo
                        //desenhada.
    refOwner                                    : TComponent;
          pontoBase                             : TPoint;

    widthAnterior                           : Integer;                          //  Usado para o
redimencionamento
    heightAnterior                          : Integer;                          //dos objetos gráficos.

    widthOriginal                           : Integer;                          //          largura e comprimento
da imagem
    heightOriginal                          : Integer;                          //original.

    fShowCurrentPosition            : boolean;
                fImageDest                                      : TImage;
    fShowMagOnCanvas                        : Boolean;
    imMagnify                                   : TImage;
    fMagnifyLevel                           : Integer;
    fmagnifyWidth                           : integer;
    destroyEdit                                 : bool;
          texto                                 : TEdit;
    //objetos externos
    fDestinationPanel               : TStatusPanel;
    fImageNumber                            : TStatusPanel;
    fWC                                         : TEdit;
```

```
    fWW                                                    : TEdit;

    //atributos do piloto
//    pilotOrient                                          : TPilotOrient;
//    pilotRef                                             : Integer;

    //atributo para alteracao de window
    alterandoWindow                          : boolean;

////////////////////////////////////////
    ImageDesignHeight:  INTEGER;
    ImageDesignWidth :  INTEGER;
    MagnifierShowing :  BOOLEAN;
    BackupBitmap: TBitmap; // [anme]

////////////////////////////////////////
    function calculaExpansaoX() : real;
    function          calculaExpansaoY() : real;
    PROCEDURE WmEraseBkgnd(VAR Msg:  TWmEraseBkgnd); MESSAGE Wm_EraseBkgnd;
  public
    redesenhar                                             : boolean;

    constructor Create(AOwner : TComponent); override;
    destructor  Destroy; override;

    procedure         atualizaDesenho();
    procedure         reposiciona();
    procedure         desenhaSelecao();
    procedure         setaImagem(im : TCyclopsImage);
    procedure         setaBorda(Borda : Bool);
    procedure         setaEspessuraBorda(espessura : Integer);
    procedure         desenhaBorda;

    procedure         escreveNumero;

    function mergeScheme:TBitmap;
    procedure         rotate(orientation: OrientationType);

    procedure         MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y: Integer); override;
    procedure         MouseMove(Shift: TShiftState; X, Y: Integer); override;
    procedure         MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer); override;
    procedure         Paint; override;
    procedure         Resize; override;

    procedure         desenhaRetangulo(x,y : Integer);
    procedure         desenhaRetanguloArredondado(x,y : Integer);
    procedure         desenhaElipse(x,y : Integer);
    procedure         desenhaCirculo(x,y : Integer);
    procedure         desenhaLinha(x,y : Integer);
    procedure         selecionaFigura(x,y : Integer;Shift: TShiftState);
    procedure         desenhaTexto(x,y : Integer);
    procedure         apagaSelecionados;
    procedure         apagaTodos;

    procedure         mostraLupa(x,y : Integer);
    procedure         setaLupa;

    procedure         changePalette;
    procedure         alteraWindow(x,y : integer);
  published
      property        image                : TCyclopsImage read imagemDeFundo write setaImagem;
      property        tool                 : TDrawingTool read ferramenta write ferramenta;
      property        beBordered  : Bool read comBorda write setaBorda;
      property        borderColor          : TColor read  corDaBorda write corDaBorda;
      property  borderWidth : Integer read espessuraDaBorda write setaEspessuraBorda;

      property        ShowCurrentPosition : boolean
              read    fShowCurrentPosition
        write fShowCurrentPosition
        default false;
```

```
    property            showImageNumber : boolean
                read    fShowImageNumber
      write fShowImageNumber
      default true;

    property            DestinationPanel : TStatusPanel
                read    fDestinationPanel
      write fDestinationPanel;
    property            ImageNumber : TStatusPanel
                read    fImageNumber
      write fImageNumber;
    property            refWW                       : TEdit
                read    fWW
      write   fWW;
    property            refWC                       : TEdit
                read    fWC
      write   fWC;

    property            MagnifyLevel : Integer
                read    fMagnifyLevel
      write fMagnifyLevel
      default 2;
    property            magnifyWidth : Integer
                read    fmagnifyWidth
                write   fmagnifyWidth
      default           100;
    property            ShowMagOnCanvas : Boolean
                read    fShowMagOnCanvas
      write fShowMagOnCanvas
      default false;
    property            ImageDest : TImage
                read    fImageDest
      write fImageDest;

end;
implementation
{-------------------------------------------------------------------------------}
constructor TPaintBoxEx.Create(AOwner : TComponent);
begin
   redesenhar := false;
   //cria uma lista de figuras que conterá todos os objetos gráficos desenhados
   listaDeFiguras := TVectorGraphicsList.Create;
            inherited Create(AOwner);

   estadoDoDesenho := dsNotDrawing;
   refOwner := AOwner;
   corDaBorda := clRed;
   espessuraDaBorda := 1;
   contResize := 1;

   //cria a lupa
   imMagnify := TImage.Create(AOwner);
   imMagnify.Parent := AOwner as TWinControl;
   imMagnify.Visible := false;
//   imMagnify.Stretch := true;
   MagnifyLevel := 2;
   magnifyWidth := 200;

   destroyEdit := false;
   showImageNumber := true;
end;
{-------------------------------------------------------------------------------}
destructor TPaintBoxEx.Destroy;
begin
            inherited Destroy;

   listaDeFiguras.Free;
   imMagnify.Free;
   imMagnify := nil;
end;
```

```
{-------------------------------------------------------------------------------}
{Indica para o componente que ele deve se posicionar sobre a imagem}
procedure  TPaintBoxEx.reposiciona();
begin
   widthAnterior       := Width;
   heightAnterior      := Height;

            top := imagemDeFundo.Top;
   Width := imagemDeFundo.Width;
   Left := imagemDeFundo.Left;
   Height := imagemDeFundo.Height;


end;
{-------------------------------------------------------------------------------}
//seleciona os objetos
procedure TPaintBoxEx.desenhaSelecao;
var
            cont        : Integer;
   figura   : TVectorGraphicsNode;
begin
            for cont := 0 to listaDeFiguras.Count -1 do
   begin
                        figura := listaDeFiguras[cont];
      if figura.Selected then
      BEGIN
            figura.writeArea := TRUE;
            figura.DrawHandles(Canvas,clAqua,clAqua,4);
      END;
   end;
   Paint;

end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.setaImagem(im : TCyclopsImage);
begin
   imagemDeFundo := im;
   reposiciona;
   widthOriginal  := imagemDeFundo.Width;
   heightOriginal := imagemDeFundo.Height;

end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.setaBorda(Borda : Bool);
begin
   comBorda := Borda;
   Refresh;
   desenhaBorda;


end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.setaEspessuraBorda(espessura : Integer);
begin
   espessuraDaBorda := espessura;
   Refresh;
   desenhaBorda;
end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaBorda;
var
            corAntiga                                    : TColor;
  espessuraAntiga          : Integer;
  offSet                                                 : integer;
begin
   if comBorda then
   begin
            Canvas.brush.color := clNone;
               Canvas.brush.Style := bsClear;
      Canvas.pen.mode := pmCopy;
```

```
      espessuraAntiga := Canvas.pen.Width;
      corAntiga := Canvas.Pen.Color;

      Canvas.pen.Width := espessuraDaBorda;
      Canvas.Pen.Color := corDaBorda;

      offSet := espessuraDaBorda mod 2;
      Canvas.Rectangle(offSet, offSet, width - offSet, height - offSet);

      Canvas.Pen.Color := corAntiga;
            Canvas.pen.Width := espessuraAntiga;

   end;
end;
{-------------------------------------------------------------------------}
////////////////////EVENTOS SOBRESCRITOS DE TPaintBox//////////////////////
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
   Hide;

            //define thew current image
   ImageNumber.Text := IntToStr(imagemDeFundo.imageNumber);

            pontoBase := Point(x,y);
   Canvas.pen.mode := pmNotXor;
   Canvas.brush.color := clNone;
   Canvas.brush.Style := bsClear;
   Screen.Cursor := crCross;
   RestrictCursorToDrawingArea(self);

   if destroyEdit then
   begin
                        desenhaTexto(x,y);
   end;

            //caso o botao esquedo do mouse tenha sido clicado
            if button = mbLeft then
   begin
            case ferramenta of
            dtRectangleTool                 : desenhaRetangulo(x,y);
         dtRoundRectangleTool    : desenhaRetanguloArredondado(x,y);
         dtEllipseTool           : desenhaElipse(x,y);
         dtCircleTool                       : desenhaCirculo(x,y);
         dtLineTool                                 : desenhaLinha(x,y);
         dtSelectTool               : selecionaFigura(x,y,Shift);
//          dtTextTool                                : desenhaTexto(x,y);
         dtMagnify                                  : setaLupa;
         dtWindowTool                               : begin
                                                                                alteraWindow(x,y);
                              alterandoWindow := true;
                             end;
      else
      end;
      if ferramenta = dtTextTool then
      begin
         destroyEdit := true;

         texto := TEdit.Create(Owner);
         texto.Parent := Parent;
         texto.Top := y;
         texto.Left := x;
         texto.SetFocus;

         //seta a ferramenta default
         tool := dtSelectTool;
      end;
   end;
```

```
    //caso o botao direito do mouse tenha sido clicado
    if button = mbRight then
    begin
    end;
end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.MouseMove(Shift: TShiftState; X, Y: Integer);
var
        TranslateVector : TPoint;
begin
        if ferramenta = dtMagnify then
    begin
                    IF   MagnifierShowing then
            mostraLupa(x,y);
       exit;
    end;
    if ferramenta = dtWindowTool then
    begin
      if alterandoWindow then
            alteraWindow(x,y);
    end;
    if estadoDoDesenho = dsNewFigure then
    begin
        Canvas.Brush.Style := bsClear;
      Canvas.Pen.Mode := pmNotXor;

      /////////Erase//////////
       //apaga a figura anterior
        figuraCorrente.DrawFigure(Canvas);

          /////////draw////////
      //desenha a nova figura
                figuraCorrente.PointB := Point(x,y);
        figuraCorrente.DrawFigure(Canvas);


      Canvas.Pen.Mode := pmCopy;
    end;

   if (estadoDoDesenho = dsTranslate) then
   begin
                // XOR to remove old figure
      Canvas.Pen.Mode := pmNotXOR;
      listaDeFiguras.DrawSelectedFigures(Canvas);

      // draw figure at new position
      TranslateVector := SubtractPoints(Point(X,Y), PontoBase);
      listaDeFiguras.TranslateSelectedFigures(TranslateVector);
      listaDeFiguras.DrawSelectedFigures(Canvas);

       //DrawingBasePoint := NewPoint;
      PontoBase := Point(X,Y);
      Canvas.Pen.Mode := pmCopy
   end;

   ///show current position on referenced statusPanel
   DestinationPanel.Text := IntToStr(X) + '@' + IntToStr(Y);

   //show current HU value at mouse position
   //DestinationPanel.Text := inttostr(imagemDeFundo.pixelValue[y,x]);;

end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    if ferramenta = dtWindowTool then
        alterandoWindow := false;

        if ferramenta = dtMagnify then
```

```
    begin
      MagnifierShowing := FALSE;
      Screen.Cursor := crDefault;
      imagemDeFundo.Picture.Graphic  := BackupBitmap;  // Restore base image // [anme]
     if (BackupBitmap <> nil) then // [anme]
                BackupBitmap.Free;
    end;
      //imMagnify.Visible := false;
    if estadoDoDesenho = dsNewFigure then
    begin

      estadoDoDesenho := dsNotDrawing;
      figuraCorrente.PointB := Point(x,y);
      figuraCorrente.writeArea := true;

      if ferramenta <> dtLineTool then //no caso da linha pode ser dos dois lados
                figuraCorrente.StandardizeOrder;

      listaDeFiguras.SetSelectedFlags(False);
      listaDeFiguras.Add(figuraCorrente);
      listaDeFiguras.SetSelectedIndex(listaDeFiguras.Count- 1);
      listaDeFiguras.DrawAllFigures(Canvas);


      figuraCorrente := nil;

    end;
    if estadoDoDesenho = dsTranslate then
    begin
              RePaint;
            listaDeFiguras.DrawSelectedFigures(Canvas);
      desenhaSelecao;
    end;


    Screen.Cursor := crDefault;
    estadoDoDesenho := dsNotDrawing;
    RemoveCursorRestrictions;

    Show;

end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.Paint;
begin

    listaDeFiguras.DrawAllFigures(Canvas);
    desenhaBorda;
    escreveNumero;


end;
{-------------------------------------------------------------------------------}
//este método é invocado toda vez que o componente é redimencionado
procedure TPaintBoxEx.Resize;
begin
    inherited Resize;
    if contResize = 2 then
    begin
              atualizaDesenho;
       contResize := 1;
    end else
    if contResize = 1 then
              contResize := 2;

              //desenhaSelecao;
end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaRetangulo(x,y : Integer);
begin
```

```
            {Canvas.pen.mode := pmNotXor;
    Canvas.brush.color := clNone;
    Canvas.brush.Style := bsClear;}

            figuraCorrente := TRectangleNode.Create(self.Canvas.pen.Color,
                                                    self.Canvas.Pen.Style,
                    self.Canvas.Pen.Width,
                    Point(x,y), Point(x,y));

    estadoDoDesenho := dsNewFigure;
end;

{-------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaRetanguloArredondado(x,y : Integer);
begin
            figuraCorrente := TRoundRectangleNode.Create(self.Canvas.pen.Color,
                                                    self.Canvas.Pen.Style,
                    self.Canvas.Pen.Width,
                    Point(x,y), Point(x,y));

    estadoDoDesenho := dsNewFigure;

end;
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaElipse(x,y : Integer);
begin
            figuraCorrente := TEllipseNode.Create(self.Canvas.pen.Color,
                                                    self.Canvas.Pen.Style,
                    self.Canvas.Pen.Width,
                    Point(x,y), Point(x,y));

    estadoDoDesenho := dsNewFigure;


end;
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaCirculo(x,y : Integer);
begin
            figuraCorrente := TCircleNode.Create(self.Canvas.pen.Color,
                                                    self.Canvas.Pen.Style,
                    self.Canvas.Pen.Width,
                    Point(x,y), Point(x,y));

    estadoDoDesenho := dsNewFigure;


end;
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaLinha(x,y : Integer);
begin
            figuraCorrente := TLineNode.Create(self.Canvas.pen.Color,
                                                    self.Canvas.Pen.Style,
                    self.Canvas.Pen.Width,
                    Point(x,y), Point(x,y));

    estadoDoDesenho := dsNewFigure;


end;
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.desenhaTexto(x,y : Integer);
begin
            figuraCorrente := TTextNode.Create(self.Canvas.pen.Color,
                                                    self.Canvas.Pen.Style,
                    self.Canvas.Pen.Width,
                    Point(texto.Left,texto.Top),
                    Point(texto.Left + texto.Width,texto.Top + texto.Height),
                    texto.Text);
    texto.Destroy;
    destroyEdit := false;
```

```
    estadoDoDesenho := dsNotDrawing;

//    figuraCorrente.writeArea := true;


    listaDeFiguras.SetSelectedFlags(False);
    listaDeFiguras.Add(figuraCorrente);
    listaDeFiguras.SetSelectedIndex(listaDeFiguras.Count - 1);
    listaDeFiguras.DrawAllFigures(Canvas);


    figuraCorrente := nil;

    Screen.Cursor := crDefault;
    estadoDoDesenho := dsNotDrawing;
    RemoveCursorRestrictions;


end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.selecionaFigura(x,y : Integer;Shift: TShiftState);
begin
            listaDeFiguras.SelectFigures(Shift,x,y);

    if (listaDeFiguras.SelectedFigureCount > 0 ) then
    begin
                desenhaSelecao;
                estadoDoDesenho := dsTranslate;
    end else
    begin

            if (comBorda) then
            setaBorda(false)
            else
            setaBorda(true)
    end;
end;
{-------------------------------------------------------------------------------}
//apaga todas as figuras que estao selecionadas
procedure TPaintBoxEx.apagaSelecionados;
begin
            listaDeFiguras.DeleteSelectedFigures;
    RePaint;
end;
{-------------------------------------------------------------------------------}
{                                                                    Métodos Privados
                                                }
{-------------------------------------------------------------------------------}
function    TPaintBoxEx.calculaExpansaoX : real;
begin
    //origemCorrente   : TPoint;
    //destinoCorrente  : TPoint;

    result := Width / widthAnterior;

end;
{-------------------------------------------------------------------------------}
function    TPaintBoxEx.calculaExpansaoY : real;
begin
            result := Height / heightAnterior;
end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.atualizaDesenho;
begin
    if redesenhar then
    begin
       listaDeFiguras.ScaleAllFigures(calculaExpansaoX,calculaExpansaoY);
       listaDeFiguras.DrawAllFigures(Canvas);

       heightAnterior := Height;
```

```
      widthAnterior := Width;
            end;
end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.apagaTodos;
begin
            listaDeFiguras.Clear;
   RePaint;
end;
{-------------------------------------------------------------------------------}
function   TPaintBoxEx.mergeScheme: TBitmap;
var
            mergedBitmap          : TBitmap;
begin
   //cria e dimenciona o bitmap original
   mergedBitmap := TBitmap.Create;
   mergedBitmap.Width := widthOriginal;
   mergedBitmap.Height := widthOriginal;
   clipboard.Assign(imagemDeFundo.Picture.Bitmap);
   mergedBitmap.Assign(clipboard);
   mergedbitmap.Canvas.CopyMode := cmMergeCopy;
   listaDeFiguras.DrawAllFigures(mergedBitmap.Canvas);

   mergeScheme := mergedBitmap;

end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.rotate(orientation: OrientationType);
var
            bmp : TBitmap;
begin
            bmp := Tbitmap.Create;
   bmp.Assign(imagemDeFundo.Picture.Bitmap);

            if ((orientation = Ottop) or (orientation = Otbottom)) then
      imagemDeFundo.Picture.Bitmap.Assign(flip(bmp));

   if (orientation = OtLeft) then
      imagemDeFundo.Picture.Bitmap.Assign(RotateLeft(bmp));

            if (orientation = Otright) then
      imagemDeFundo.Picture.Bitmap.Assign(RotateRight(bmp));

end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.mostraLupa(x,y : Integer);
var
   AreaRadius    :  INTEGER;
            Magnification :  INTEGER;
   ModifiedBitmap:  TBitmap;
   xActual       :  INTEGER;
   yActual       :  INTEGER;

{          recS        : Trect;
   recD     : TRect;
   zoom     : integer;
}
begin
   xActual := MulDiv(X, imagemDeFundo.Picture.Bitmap.Width,  imagemDeFundo.Width);
   yActual := MulDiv(Y, imagemDeFundo.Picture.Bitmap.Height, imagemDeFundo.Height);

   Magnification := MagnifyLevel;
            AreaRadius := ROUND(magnifyWidth / Magnification);

   ModifiedBitmap := TBitmap.Create;
   WITH ModifiedBitmap DO
   BEGIN
               Assign(BackupBitmap);   // Make acopy of the "base" image // [anme]

      // Single-pixel border when requested
```

```
      Canvas.Brush.Color := clBlack;
//      Canvas.Pen.Color   := ColorDialog.Color;
      Canvas.Pen.Style   := psSolid;
      // Outline for magnifier to help contrast between magnifier
      // and any image.
      Canvas.Rectangle (xActual - AreaRadius * Magnification-1,
                        yActual - AreaRadius * Magnification-1,
                        xActual + AreaRadius * Magnification+1,
                        yActual + AreaRadius * Magnification+1);

      Canvas.CopyMode := cmSrcCopy;
      Canvas.CopyRect(Rect(xActual - AreaRadius * Magnification,
                        yActual - AreaRadius * Magnification,
                        xActual + AreaRadius * Magnification,
                        yActual + AreaRadius * Magnification),
                      BackupBitmap.Canvas, // [anme]
                      Rect(xActual - AreaRadius,
                        yActual - AreaRadius,
                        xActual + AreaRadius,
                        yActual + AreaRadius) );
      // Display newly modified image
                imagemDeFundo.Picture.Graphic := ModifiedBitmap;
              ModifiedBitmap.Free;

      if ImageDest <> nil then
                ImageDest.Canvas.CopyRect(rect(0,0,198,218),
                                                        imagemDefundo.Picture.Bitmap.Canvas,
                        Rect(xActual - AreaRadius,
                        yActual - AreaRadius,
                        xActual + AreaRadius,
                        yActual + AreaRadius) );
   END;

////////////////////////////////////////////////////////////////////////////
{    //LockWindowUpdate(Parent.Handle);
   if imMagnify.Visible = false then
              exit;

//    case MagnifyLevel of
//       mag100          : zoom := magnifyWidth div 2;
//       mag200          : zoom := magnifyWidth div 4;
//       mag400          : zoom := magnifyWidth div 8;
//    else
//       zoom:= -1;

//    end;

    //rectangle source
   if ((X - (zoom)) < 0) or ((Y - (zoom))< 0) then
              exit;
   if ((X + (zoom)) > imagemDefundo.Width ) or ((X + (zoom)) > imagemDefundo.Height) then
              exit;

    recS.Left := abs(X - (zoom ));
    recS.Top := abs(Y - (zoom ));
    recS.Right := abs(X + (zoom ));
   recS.Bottom := abs(Y + (zoom));
    //rectangle destination
   recD.Top := 0;
   recD.Left := 0;
   recD.Right := magnifyWidth;
   recD.Bottom := magnifyWidth;

//    imMagnify.Left := ((X - (magnifyWidth div 2))) + imagemDeFundo.Left;
//    imMagnify.Top := ((Y  - (magnifyWidth div 2)))+ imagemDeFundo.top;
    imMagnify.Left := ((X - (magnifyWidth div 2))) + imagemDeFundo.Left;
   imMagnify.Top := ((Y - (magnifyWidth div 2)))+ imagemDeFundo.top;

    imMagnify.Width := magnifyWidth;
   imMagnify.Height := magnifyWidth;
```

```
    //imMagnify.Canvas.CopyRect(recD,imagemDefundo.Canvas,recS);
    imMagnify.Canvas.CopyRect(recD,imagemDefundo.Canvas,recS);
            imMagnify.Canvas.FrameRect(recD);
    imMagnify.Picture.Bitmap.Canvas.FrameRect(recD);

    if ImageDest <> nil then
            ImageDest.Canvas.CopyRect(rect(0,0,magnifyWidth,magnifyWidth),
                                                    imagemDefundo.Picture.Bitmap.Canvas,recS);
}
end;
{-------------------------------------------------------------------------------}
procedure TPaintBoxEx.setaLupa;
begin
            BackupBitmap := TBitmap.Create;
            //Save original image
            BackupBitmap.Assign(imagemDeFundo.Picture.Bitmap);
    MagnifierShowing := true;
{           if imMagnify.Width <> magnifyWidth then
    begin
      //deleta a lupa antiga
      imMagnify.Free;
      imMagnify := nil;
      //cria a lupa
      imMagnify := TImage.Create(refOwner);
      imMagnify.Parent := refOwner as TWinControl;
      imMagnify.Width := magnifyWidth;
      imMagnify.Height := magnifyWidth;
      imMagnify.Visible := true;
      imMagnify.BringToFront;
    end else
    begin
                        imMagnify.Visible := true;
      imMagnify.BringToFront;
    end;
}
end;
{-------------------------------------------------------------------------------}
//description:  Esta ferramenta muda os valores de window da imagem corrente
//e chama o método changePalette da classe CyclopsImage que altera efetivamente
//os valores dos pixels.
procedure TPaintBoxEx.alteraWindow(x,y : integer);
var
            variaW,
    variaC    : Integer;
begin
            //calcula a variacao do window center e width
    variaW := pontoBase.X - x;
    variaC := pontoBase.Y - y;
    //valida a variacao WC e WW com os valores atuais
            //se o movimento em X é positivo
    if x > pontoBase.X then
                        variaW := imagemDeFundo.WindowWidth  + variaW;
      //se o movimento em X é negativo
            if x <= pontoBase.X then
                        variaW := imagemDeFundo.WindowWidth  + variaW;
      //se o movimento em Y é positivo
            if y > pontoBase.y then
                        variaC := imagemDeFundo.WindowCenter + variaC;
                        //se o movimento em Y é negativo
            if y <= pontoBase.y then
                        variaC := imagemDeFundo.WindowCenter + variaC;

    //chama a funcao para alteraçao do window
    if variaW <= 0 then variaW := 1;
    imagemDeFundo.changePalette(variaC,variaW);
    pontobase.X := x;
    pontoBase.Y := y;

            //mostra o novo window na tela
```

```
   fWW.Text := IntToStr(variaW);
   fWC.Text := IntToStr(variaC);
end;
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.changePalette;
var
//  hpal      : HPALETTE;
  i                                   : Integer;
  pe                        : PALETTEENTRY;
  pal                       : PLogPalette;
//  bmpTemp             : TBitmap;
begin
//  pal := nil;
  try
              //criar o bitmap temporário
//            bmpTemp := imagemDeFundo.Bitmap;
    //atribuir a este bitmap o bitmap original (ler do arquivo temp)

    GetMem(pal, sizeof(TLogPalette) + sizeof(TPaletteEntry) * 255);
    pal.palVersion := $300;
    pal.palNumEntries := 256;
    for i := 0 to 255 do
    begin
              GetPaletteEntries(imagemDeFundo.Picture.Bitmap.Palette,i,1,pe);
      //cor := RGB(pe.peRed,pe.peGreen,pe.peBlue);
      //
      //processamento para ver se a entrada da paletta esta dentro do window
      // se nao setar para 0 ou 255

      //  image2.Canvas.Pen.Color := cor;
      //  image2.Canvas.Brush.Color := cor;
      //  pal.palPalEntry[i].peRed :=
      //  pal.palPalEntry[i].peGreen :=
      //  pal.palPalEntry[i].peBlue :=
    end;
    //hpal := CreatePalette(pal^);
    ///if hpal <> 0 then
    //begin
                        //atribuir o bitmap temporário para imagemDeFundo
                        //imagemDeFundo.picture.Bitmap.Palette := hpal;
    //end;
  finally
    //FreeMem(pal);
  end;
end;
{-------------------------------------------------------------------------}
procedure TPaintBoxEx.escreveNumero;
begin
  if showImageNumber then
  begin
    Canvas.brush.color := clNone;
    Canvas.brush.Style := bsClear;
    Canvas.pen.mode := pmCopy;

    with Canvas.Font do
    begin
              Color := clred;
      Size := 11;
      Style := [fsBold];
    end;

              Canvas.TextOut(self.Width - 20, self.Height -20,IntToStr(imagemDeFundo.imageNumber));
  end;

end;
{-------------------------------------------------------------------------}
//description:  Esta ferramenta muda os valores de window da imagem corrente
//e chama o método changePalette da classe CyclopsImage que altera efetivamente
//os valores dos pixels.
//procedure TPaintBoxEx.alteraWindow(x; y);
```

```
//var
//              variaW
//    variaC                : INteger;
//begin
//
//
//end;
{------------------------------------------------------------------------------}
PROCEDURE TPaintBoxEx.WmEraseBkgnd(VAR Msg:  TWmEraseBkgnd);
BEGIN
  Msg.Result := 1;
END {WmEraseBkgnd};
end.

//-----------------------------------------------------------------------------------------------------------

unit uPaperSheet;

interface

uses extCtrls, controls;

type TPaperPicture = record
  left, top: integer;
  width, height: integer;
  slot: TPanel;
  image: TImage;
end;

type TPaperModel = class
protected
  fimgsPerRow: integer;
  procedure setImgsPerRow(aValue: integer);
  procedure refreshSlotsPos;
public
  pictures: array [1..4, 1..8] of TPaperPicture;
  paperModel: TPanel;
  ImagesWidth, ImagesHeight: integer;
  slotsWidth, slotsHeight: integer;
  constructor createOn(aPaperModel: TPanel; imWidth, imHeight: integer);
  procedure activate;
  procedure enableSlot(x,y: integer; left, top: integer);
  function getMaxRows: integer;
  procedure getSlotsDimensions;
  property imgsPerRow: integer read fimgsPerRow write setImgsPerRow;
end;

implementation

{ TPaperModel }
//------------------------------------------------------------------------------
procedure TPaperModel.activate;
var
  i, j: integer;
begin
  for i:= 1 to 4 do
    for j:= 1 to 8 do
      pictures[i, j].slot.parent:= paperModel;
end;
//------------------------------------------------------------------------------
constructor TPaperModel.createOn(aPaperModel: TPanel; imWidth, imHeight: integer);
var
  i, j: integer;
begin
  paperModel:= aPaperModel;
  imagesWidth:= imWidth;
  ImagesHeight:= imWidth;
  for i:= 1 to 4 do
    for j:= 1 to 8 do
      with pictures[i, j] do
```

```
      begin
      left:= 0;
      top:= 0;
      width:= 50;
      height:= 50;
      slot:= TPanel.Create(paperModel);
      slot.Visible:= false;
      image:= TImage.Create(slot);
      image.Align:= alClient;
      image.Stretch:= true;
      image.Visible:= true;
      image.parent:= slot;
      end;
end;
//---------------------------------------------------------------------------
procedure TPaperModel.enableSlot(x, y, left, top: integer);
begin

end;
//---------------------------------------------------------------------------
function TPaperModel.getMaxRows: integer;
begin

end;
//---------------------------------------------------------------------------
procedure TPaperModel.getSlotsDimensions;
begin

end;
//---------------------------------------------------------------------------
procedure TPaperModel.refreshSlotsPos;
begin

end;
//---------------------------------------------------------------------------
procedure TPaperModel.setImgsPerRow(aValue: integer);
begin

end;
//---------------------------------------------------------------------------
end.

//-----------------------------------------------------------------------------------------------------------

unit uThumbnails;

interface

uses menus, graphics, forms, classes, extCtrls, uCyclopsImage, uMiniView;

type TThmubnailsManager = class

protected
  scrollbox: TScrollbox;
  popup: TPopupMenu;
public
  thumbs: TList;
  constructor createOn(aScrollBox: TScrollbox; popupMenu: TPopupMenu);
  procedure clearThumbs;
  procedure addImage(aCyclopsImage: TCyclopsImage);
  procedure imageClick(Sender: TObject);
  procedure imageDClick(Sender : TObject);
end;

type TThumbnail = class
protected
  fLeft, fTop: integer;
  selected: boolean;
  procedure setLeft(aValue: integer);
  procedure setTop(aValue: integer);
```

```
public
  cycImage: TCyclopsImage;
  Image: TCyclopsImage;
  mark: TShape;
  manager: TThmubnailsManager;
  constructor createWith(acycImage: TCyclopsImage; aManager: TThmubnailsManager);
  procedure select;
  procedure unselect;
  procedure activate;
  procedure freeSubcomponents;
  property Left: integer read fLeft write setLeft;
  property top: integer read ftop write setTop;
end;

implementation

uses Controls;

{ TThumbnail }
//---------------------------------------------------------------------------
procedure TThumbnail.activate;
begin
  Image.Visible:= true;
end;
//---------------------------------------------------------------------------
constructor TThumbnail.createWith(acycImage: TCyclopsImage; aManager: TThmubnailsManager);
var
  bmp: TBitmap;
  size: integer;
begin
  cycImage:= aCycImage;
  manager:= aManager;
  size:= manager.scrollbox.Height - 28;
  bmp := cycImage.bitmap;
  image:= TCyclopsImage.Create(manager.scrollbox);
  with image do
    begin
    Stretch := true;
    Picture.Bitmap.Assign(bmp);
    Width := size;
    Height := size;
    PopupMenu:= manager.popup;
    Visible:= false;
    Parent := manager.scrollbox;
    OnClick:= manager.imageClick;
    OnDblClick := manager.imageDClick;
    end;
  mark:= TShape.Create(manager.scrollbox);
  with mark do
    begin
    Width:= size + 4;
    Height:= size + 4;
    Brush.Color:= clRed;
    SendToBack;
    Visible:= false;
    Parent:= manager.scrollbox;
    Pen.Color:= clRed;
    Brush.Style:= bsClear;
    pen.Width:= 3;
    SendToBack;
    end;
  left:= 0;
  top:= 0;
end;
//---------------------------------------------------------------------------
procedure TThumbnail.freeSubcomponents;
begin
  Image.Free;
  mark.Free
end;
```

```
//-------------------------------------------------------------------------------
procedure TThumbnail.select;
begin
  selected:= true;
  mark.Visible:= true;
end;
//-------------------------------------------------------------------------------
procedure TThumbnail.setLeft(aValue: integer);
begin
  image.Left:= aValue;
  mark.Left:= aValue - 2;
end;
//-------------------------------------------------------------------------------
procedure TThumbnail.setTop(aValue: integer);
begin
  image.Top:= aValue;
  mark.Top:= aValue - 2;
end;
//-------------------------------------------------------------------------------
procedure TThumbnail.unselect;
begin
  selected:= false;
  mark.Visible:= false;;
end;
//-------------------------------------------------------------------------------
{ TThmubnailsManager }
procedure TThmubnailsManager.addImage(aCyclopsImage: TCyclopsImage);
var
  posX: integer;
  thumb: TThumbnail;
begin
  posX:= ((scrollbox.Height - 23) * thumbs.count) + 5;
  thumb:= TThumbnail.createWith(aCyclopsImage, self);
  thumb.Left:= posX;
  thumb.top:= 5;
  thumb.activate;
  thumbs.Add(thumb)
end;
//-------------------------------------------------------------------------------
procedure TThmubnailsManager.clearThumbs;
var
  i: integer;
  t: TThumbnail;
begin
  for i:= 0 to thumbs.Count - 1 do
    begin
    t:= thumbs.items[i];
    t.freeSubcomponents;
    t.Free;
    end;
  thumbs.Count:= 0;
end;
//-------------------------------------------------------------------------------
constructor TThmubnailsManager.createOn(aScrollBox: TScrollbox; popupMenu: TPopupMenu);
begin
  scrollbox:= aScrollBox;
  popup:= popupMenu;
  thumbs:= TList.Create;
end;
//-------------------------------------------------------------------------------
procedure TThmubnailsManager.imageClick(Sender: TObject);
var
  i: integer;
  th: TThumbnail;
begin
  th := nil;
  for i:= 0 to thumbs.count- 1 do
    begin
    th:= thumbs.items[i];
    if sender = th.Image then
```

```
      break;
    end;
  if th <> nil then
    if th.selected then
      th.unselect
    else
      th.select;
end;
//-------------------------------------------------------------------------------
procedure TThmubnailsManager.imageDClick(Sender: TObject);
var
        mv      : TMiniView;
//    refIm   : TCyclopsImage;
    i               : integer;
    th              : TThumbnail;

begin
        th := nil;
        for i:= 0 to thumbs.count - 1 do
    begin
                th:= thumbs.items[i];
        if sender = th.Image then
                break;
        end;
      if th <> nil then
      begin
        mv := TMiniView.Create(scrollbox.Owner);
        mv.im_miniView.Picture.Bit map.assign(th.cycImage.Bitmap);
        mv.im_miniView.Canvas.Pen.Color := clred;
        mv.Width :=  th.cycImage.originalWidth;
        mv.Height := th.cycImage.originalHeight;
        mv.Show;
      end;
end;
//-------------------------------------------------------------------------------
end.
```

19 Bibliografia

Charles Petzold, Programming Windows – The definitive guite to win32 API, Microsoft Press, Washington, 1998.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 1: Introduction and Overview**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/2000/draft/00_01dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 2: Conformance**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/2000/draft/ 00_02dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) Part 3: Information Object Definitions**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/ dicom/2000/draft/00_03dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/ 2000/draft/00_05dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 6: Data Dictionary**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/2000/draft/00_06dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 10: Media Storage and File Format for Media Interchange**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/2000/draft/00_10dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 11: Media Storage Application Profiles**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/dicom/ 2000/draft/00_11dr.pdf>. Acesso em: 10 de novembro de 2000.

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **Digital Imaging and Communications in Medicine (DICOM) - Part 12: Media Formats and Physical Media for Media Interchange**. Virginia, 2000. Disponível em: <ftp://medical.nema.org/medical/ dicom/2000/draft/00_12dr.pdf>. Acesso em: 10 de novembro de 2000.

DELLANI, Paulo - **Desenvolvimento de um Servidor de Imagens Médicas Digitais no Padrão Dicom.** Dissertação de Mestrado - 2001

COSTA SAMPAIO, Silvio – **Modelagem e implementação orientada a objetos de um cliente de rede para banco de dados de imagens médicas digitais utilizando o padrão DICOM 3.0**. Dissertação de Mestrado - 1999

NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION. **NEMA's OFFICIAL DICOM WEB PAGE**. Disponível em: <http://medical.nema.org/dicom.html>. Acesso em: 20 de novembro de 2000.

THE CYCLOPS PROJECT, .Disponível em <http:/www.inf.ufsc.br/cyclops>. Acesso em 11/12/2001

CLUNIE, David A. **Medical Image Format FAQ**. Questões freqüentemente perguntadas sobre Formatos de Imagens Médicas, com enfoque para o padrão DICOM. Disponível em: <http://www.dclunie.com/medical-image-faq/html/>. Acesso em: 11 de dezembro de 2001.