UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM AMBIENTE PARA TRATAMENTO DE IMAGENS *RASTER* 2D

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ADRIANA FORNAZARI

BLUMENAU, NOVEMBRO/2000

2000/2-02

PROTÓTIPO DE UM AMBIENTE PARA TRATAMENTO DE IMAGENS *RASTER* 2D

ADRIANA FORNAZARI

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Dalton Solano dos Reis — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Dalton Solano dos Reis

_____

Prof. Paulo César Rodacki Gomes

_____

Prof. Maurício Cappobianco Lopes

DEDICATÓRIA

Dedico este trabalho a meus pais,

e ao meu namorado Angelo.

AGRADECIMENTOS

Primeiramente a Deus que sem Ele nada seria possível, me deu forças que eu não tinha e ensinou-me o caminho para chegar aqui.

Ao meu professor e orientador Dalton Solano dos Reis, pela sua dedicação e orientação na elaboração deste trabalho.

Ao meu namorado, Angelo, que me ajudou nas horas mais difíceis com sua compreensão.

Aos meus amigos e colegas da faculdade; Luciane e Daniel pelo companheirismo.

Ao amigo Oslí que sempre me ajudou quando precisei, principalmente quanto a utilização do ambiente Delphi.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

RESUMO

Este trabalho visa a realização de um estudo sobre ambientes e técnicas de processamento de imagens *raster* 2D e a especificação e implementação de um protótipo que apresente a criação de um ambiente com as técnicas de espelho, reflexo, rotação, preenchimento de regiões e realce de figuras através do seu histograma. As ferramentas utilizadas serão o *Power Designer* 6.1 para fazer a especificação do protótipo e o *Delphi* 5.0 para a sua implementação.

ABSTRACT

This work lens the accomplishment of a study about techniques and tools of *raster* 2D image processing, the specification and implementation of a prototype that will present the creation of a software tool with techniques of mirror, reflex, rotation, painting of areas and enhance of picture through their histogram. The Power Designer 6.1 tools was used in specification of prototype and the Delphi 5.0 was used in implementation.

1 INTRODUÇÃO

Quando foram criados, os computadores eram utilizados para calcular e imprimir números. Desde então sua utilização se diversificou muito. Nos últimos anos, com o surgimento de novas tecnologias, a utilização de funções gráficas se tornou indispensável para profissionais de diversas áreas, surgindo assim a área da computação gráfica ([BRO1995]).

Computação gráfica é uma vasta área do conhecimento humano, abrangendo a geração, manipulação e análise de imagens obtidas através de digitalização ou de uma descrição formal ([MAG1986]).

A área da computação gráfica é relativamente recente. Seu desenvolvimento mais acentuado iniciou-se na década de 70, com a superação de obstáculos tecnológicos através do progresso da eletrônica e, conseqüente, melhora da relação custo/benefício dos dispositivos gráficos.

Com o barateamento dos dispositivos gráficos de entrada e saída, houve um aumento da faixa de usuários, e com isso a popularização desta área de trabalho ([MAG1986]).

Para ser possível manipular ou interpretar imagens é necessária a utilização de alguma técnica de armazenamento e especificação das figuras, que torne possível sua visualização em algum dispositivo computacional ([PER1989]).

A imagem gráfica com formato *raster* é a mais comum tecnologia de representação de imagens em uso hoje, pois torna possível simular os efeitos de cor, luz e sombra nos objetos realísticos e permite manipular o menor detalhe da figura. As figuras normalmente especificadas por este meio são imagens captadas através de *scanners*, para serem modificadas através da aplicação de técnicas de manipulação de suas características originais. A esta manipulação chama-se de processamento de imagens ([CHA1989]).

O processamento de imagem envolve técnicas de transformação onde tanto a imagem de partida quanto a de resultado apresentam-se sob uma representação visual. As transformações visam possibilitar modificações nas características visuais da imagem, tais como duplicação e eliminação de pixels ou ainda reduzir ruídos e distorções ([PER1989]).

Para tais transformações se tornarem possíveis, algoritmos devem ser desenvolvidos para cada técnica de processamento de imagem que se queira implementar. Nos últimos anos

diversos fabricantes de *softwares* se dedicaram a implementar e disponibilizar para o mercado ambientes que possibilitam a utilização destas técnicas.

Para o desenvolvimento deste trabalho é necessário o uso de ferramentas de especificação e implementação. O *Power Designer* 6.1 foi utilizado na especificação e o *Delphi* 5.0 foi utilizado como ferramenta de implementação do protótipo.

1.1 OBJETIVO

Os objetivos principais deste trabalho são de estudar ambientes e técnicas de processamento de imagens para arquivos no formato *raster* 2D e a especificação e desenvolvimento de um protótipo que implemente as técnicas de espelho, reflexo, rotação, preenchimento de regiões, exibição de histograma e realce de imagens, resultando em modificações nas suas características visuais.

1.2 ORGANIZAÇÃO DO TEXTO

No capítulo dois deste trabalho serão mostrados alguns ambientes de tratamento de imagens em uso hoje, com suas características principais e em que se destacam.

O capítulo três é dedicado aos conceitos de arquivos gráficos, tecnologia *raster* e aspectos de tratamento de cores.

O capítulo de número quatro está voltado aos conceitos e as técnicas de processamento de imagens.

O capítulo cinco fala sobre as técnicas e ferramenta de especificação utilizadas na elaboração do protótipo, e mostra a especificação e suas características.

No capítulo seis são feitas avaliação do protótipo e considerações finais.

2 SISTEMAS DE TRATAMENTO DE IMAGENS

Existem diversos programas que fazem tratamento de imagens *raster* 2D. Esta seção tem o objetivo de abordar características principais de alguns destes ambientes.

2.1 FERRAMENTAS DE MANIPULAÇÃO DE IMAGENS

Como algumas ferramentas padrão ou comuns são encontradas em praticamente qualquer ambiente de edição de imagens, optou-se, primeiramente, em abordar características destas ferramentas comuns que a tecnologia dos gráficos por computador tem produzido.

2.1.1 CONCEITOS DE CORTE E COLAGEM

Como o próprio nome sugere, corta-se algo de sua posição original e cola-se em outra posição. Com a chegada das máquinas de cópia, o original nem sempre tem que sofrer corte. Pode-se fazer uma fotocópia do original, e depois colar a cópia em vez do original.

Com as interfaces gráficas disponíveis aos usuários, a analogia do cortar e colar é a mesma. Usando o *mouse* ou algum outro dispositivo de apontamento, escolhe-se alguns dados na tela e depois ativa-se geralmente o comando Cortar (no inglês *Cut*) do menu Editar (*Edit*). Para colar os dados em uma nova posição, aponta-se para a nova posição com o *mouse* e depois escolhe-se o comando Colar (*Paste*) do menu Editar. Os dados podem ser qualquer coisa – uma sentença em um processador de textos, um valor de vendas em um programa financeiro, ou os *pixels* que compõem a imagem da cabeça de uma pessoa em um programa de processamento de imagens.

Assim como no caso da máquina copiadora, o computador pode copiar os dados em vez de recortá-los da sua posição original. Escolhendo o comando Copiar (*Copy*) do menu Editar, o computador faz uma duplicata dos dados. E, então, executa-se a mesma função de colagem mostrada acima.

Uma parte importante de qualquer operação de corte ou cópia e colagem é selecionar (depois de fazer uma seleção, quaisquer alterações subseqüentes que se execute afetam apenas a área selecionada), ou seja, informar ao computador qual a parte dos dados que se deseja cortar ou copiar, fazendo assim, o seu destaque ([MOR1995]).

2.1.2 DESTAQUE

O destaque pode ser em forma de vídeo reverso, como na seleção de textos ou através de uma marca (ou *marquee* – marquise) em volta dela. Os programas gráficos freqüentemente usam uma marquise para destacar as áreas selecionadas de uma imagem, com pontos em preto-e-branco se movendo lentamente para chamar atenção para a seleção dentro da imagem. Para que o computador saiba qual a parte dos dados deve selecionar, uma de suas funções de seleção deve ser acionada ([MOR1995]).

2.1.3 SELEÇÃO DE FORMAS GEOMÉTRICAS

Embora se possa fazer conjuntos de seleção desenhando à mão livre em volta de uma certa área de uma imagem com o *mouse*, nem sempre esta é a forma mais precisa. Para resolver isso, pode-se desenhar formas geométricas. Essas formas incluem os círculos, quadrados, elipses e retângulos.

A criação de seleções geométricas envolve arrastar o *mouse*. Arrastar é um termo usado para descrever a ação de manter pressionado o botão do *mouse* enquanto move-se o *mouse*.

Para desenhar uma marquise quadrada, por exemplo, dá-se um clique no botão da marquise quadrada na caixa de ferramentas. Isso informa ao computador que se está prestes a desenhar um quadrado, e tudo o que está dentro do quadrado deve ser selecionado. A seguir move-se o *mouse* até o ponto da imagem onde se deseja posicionar o canto superior esquerdo da marquise quadrada. Depois, dá-se um clique mantendo pressionado o botão enquanto move-se o *mouse* até o ponto onde deseja-se posicionar o canto inferior direito do quadrado. Enquanto se está arrastando o *mouse*, vê-se um quadrado aumentar a partir do ponto onde se começou a arrastar o *mouse*. Quando o quadrado estiver do tamanho correto, solta-se o botão, o computador posiciona o canto inferior direito dele onde o *mouse* está localizado, criando assim, um quadrado ou retângulo.

Alguns programas de processamento de imagens oferecem duas ferramentas diferentes para selecionar quadrados ou retângulos. Outros programas possuem uma ferramenta de seleção de retângulo que pode ser transformada em um quadrado se for mantido pressionada a

tecla *shift* no teclado enquanto está criando a seleção. O programa fará a altura e a largura do retângulo do mesmo tamanho, criando um quadrado.

Outra seleção comum é o círculo. Em muitos programas de processamento de imagens, tem-se duas maneiras diferentes para desenhar círculos. Pode-se desenhar o círculo de borda a borda ou de dentro para fora. De borda a borda significa que se dá um clique uma vez para definir a borda esquerda do círculo e, depois, à medida que arrasta o *mouse*, o computador calcula um círculo com a borda esquerda no ponto onde se começou a arrastar. A borda direita do círculo permanece fixa quando se solta o botão do *mouse*. Ao desenhar o círculo de dentro para fora, o primeiro clique especifica o centro do círculo. À medida que se move o *mouse*, um círculo é computado com o seu centro baseado no ponto onde se começou a arrastar o *mouse*. A distância em que se move o *mouse* determina o tamanho do círculo ([MOR1995]).

Para selecionar áreas que não são perfeitamente circulares ou quadradas, tais como uma árvore, uma ferramenta tipo laço (*lasso*) funciona muito bem.

2.1.4 SELEÇÃO UTILIZANDO A FERRAMENTA LAÇO

A ferramenta laço é flexível e oferece métodos ou modos diferentes para obter seleções. Pode-se usar esta ferramenta de desenho à mão livre para envolver uma área na qual deseja trabalhar ou para especificar pontos individuais que o computador transforma depois em polígonos, ou a mistura dos dois.

Para desenhar utilizando esta ferramenta, enquanto o *mouse* é arrastado, o computador traça o seu caminho ao longo da tela se transformando em borda de seleção. Para fechar o caminho deve-se conectar a seleção com o ponto inicial, dando um clique duplo no *mouse*.

Para obter uma seleção de polígono, especifica-se pontos na tela e o computador conecta esses pontos. O modo como se controla o *mouse* é que determina se o laço será em forma livre ou de linhas retas ([MOR1995]).

Embora a ferramenta laço funcione bem para objetos irregulares, ela nem sempre é precisa. Para resolver esse problema, um recurso de seleção avançada chamada ferramenta caneta foi criada.

2.1.5 SELEÇÃO UTILIZANDO A FERRAMENTA CANETA

A ferramenta caneta é semelhante à ferramenta laço, exceto que ela oferece curvas muito controladas e suaves. Essas curvas, chamadas de *splines* (tiras flexíveis), fornecem um controle preciso sobre a forma da curva.

Uma *spline* é uma curva matematicamente definida. A curva em si começa a partir de um ponto de ancoragem e os seus pontos de controle. Os pontos de controle determinam a forma da curva. Pode-se assim, entortar ou moldar a curva em qualquer forma que seja necessário ajustando esses pontos de controle. Quando se move os pontos de controle para mais perto uns dos outros (em direção ou ponto de ancoragem), o ângulo fica mais agudo. Movendo os pontos de controle para mais longe uns dos outros, a curva fica maior. As *splines* funcionam bem com superfícies curvas suaves, como o rosto de uma pessoa ([MOR1995]).

Existe uma forma de seleção mais avançada e mais fácil de usar para seleções irregulares a Varinha Mágica (*Magic Wand*).

2.1.6 SELEÇÃO UTILIZANDO A FERRAMENTA VARINHA MÁGICA

Como o uso desta ferramenta pode-se selecionar qualquer *pixel* da imagem e o computador seleciona todos os *pixels* adjacentes que possuem a mesma cor, selecionando assim, rápida e precisamente, uma grande área de uma imagem com um clique.

As ferramentas de Varinha Mágica sempre oferecem uma tolerância ou uma definição de similaridade de cor. A definição de tolerância determina o quão semelhante as cores têm que ser para a seleção. Segundo Morrison ([MOR1995]) esta é uma das mais poderosas ferramentas para estabelecer seleções.

A seleção é uma parte bastante interessante no tratamento de imagens, pois através dela pode-se criar muitas cenas irreais. Para facilitar ainda mais este processo, existem outras ferramentas comuns em ambientes de tratamento de imagens utilizadas para edição. A seguir será visto algumas delas.

2.1.7 FERRAMENTA APAGAR

Todos os programas de edição de imagens possuem algum tipo de apagador (*eraser*) digital. O apagador aparece geralmente na barra de ferramentas. Os apagadores apagam a cor dos *pixels*. Para usá-los, dá-se um clique na ferramenta Apagar e o cursor se transforma no ícone do apagador. A seguir, dá-se um clique e arrasta-se o cursor ao longo da imagem e ele irá apagar quaisquer áreas sobre as quais se arraste o cursor. Basicamente, o conceito de apagar é, neste caso, nada mais que pintar *pixels* com a cor de fundo (*background*) padrão ativa no momento em que esta ferramenta é acionada. Se uma ferramenta de pintura, por exemplo, for utilizada, a cor de primeiro plano preencherá os espaços a serem pintados. Se uma ferramenta de apagador é utilizada os espaços apagados são preenchidos com a cor de fundo ([MOR1995]).

Outra maneira de cobrir erros é usando a ferramenta de borrar (*smudge*).

2.1.8 FERRAMENTA BORRAR

A ferramenta borrar (*Smudge*) é usada para misturar ou puxar cores de uma área da imagem para outra área. O computador mistura as cores observando o *mouse* quando se seleciona a ferramenta borrar. Quando se começa a arrastar o *mouse*, essa ferramenta grava as cores dos *pixels* atualmente sob o cursor. À medida que se move o *mouse*, a ferramenta passa a calcular a média dos *pixels* sobre os quais o *mouse* esta sendo movido com os *pixels* armazenados na memória ([MOR1995]).

Segundo Morrison ([MOR1995]) uma das mais poderosas ferramentas de retoque é o clone (*rubber stamp*). O uso desta ferramenta é semelhante ao uso do carimbo.

2.1.9 FERRAMENTA CLONE

Essa ferramenta copia os *pixels* de uma parte da imagem para outra parte da mesma imagem. Pode-se até copiar de uma imagem para uma figura completamente diferente.

Para usar a ferramenta clone, é necessário que primeiro se especifique um ponto-fonte para copiar os *pixels*. Esse ponto não é um conjunto de seleção; é um local em uma figura. A seguir, pode-se mover a ferramenta do clone para qualquer lugar da imagem. Quando se pressiona o botão do *mouse*, uma pequena amostra do ponto-fonte é colocada. Se o botão for

mantido pressionado a ferramenta começa a pintar em volta de uma pequena área, à medida que o *mouse* é mexido, correspondendo ao local-fonte.

Ao mesmo tempo que a ferramenta do clone está copiando pequenas seções do local-fonte, ela também está misturando as bordas dos novos *pixels* com os *pixels* existentes. Devido a essa mistura automática, a ferramenta do clone funciona bem para limpar fotografias ([MOR1995]).

Os programas de processamento de imagens oferecem funções que escurecem (*Darken*) ou clareiam (*Lighten*) todos os *pixels* sobre os quais se arraste o *mouse*.

2.1.10 FERRAMENTAS DE EMBAÇAMENTO E NITIDEZ

Estas ferramentas embaçam a imagem reunindo mais os *pixels* adjacentes ou aumentam a nitidez (ou contraste) entre os *pixels* adjacentes. A ferramenta de embaçamento (*blur*) pode ajudar a esconder pequenas falhas da imagem. A ferramenta de nitidez (*sharpen*) aumenta a nitidez das imagens que estão levemente fora de foco.

Usando as ferramentas de embaçamento e nitidez juntas, pode-se simular diferentes efeitos de câmera, tais como a profundidade do campo. Ao executar operações de corte e colagem entre imagens, a ferramenta de embaçamento pode ajudar a suavizar as bordas agudas que aparecem ([MOR1995]).

As ferramentas abordadas até este ponto fazem manipulação a nível de *pixel* (*pixel* por *pixel*) de uma imagem. Entretanto, muito pode ser feito com a manipulação de todos os *pixels* ao mesmo tempo.

Uma aplicação muito popular da edição de imagens é a correção de cores. Freqüentemente os *scanners* falham no momento de capturar toda a faixa de cores de uma imagem. Às vezes, o contraste é limitado ou os valores de brilho não são gravados com precisão. Muitos problemas de digitalização via *scanner* podem ser compensados ajustando a imagem depois que ela é digitalizada. Outras vezes, alterar a coloração de uma imagem pode ser apenas uma questão de expressão artística.

2.1.11 BRILHO E CONTRASTE

Reduzindo os valores RGB (ver seção 3.2) de um *pixel*, pode-se reduzir o brilho desses *pixels*. Da mesma forma, aumentando os valores RGB de um *pixel*, pode-se aumentar o seu brilho. Para diminuir o contraste de uma imagem, é calculada a média dos valores RGB juntos. Isso tem o mesmo efeito que reduzir a saturação de uma imagem. Para aumentar o contraste, os valores RGB são incrementados separadamente. Para fazer esses ajustes, os programas de processamento de imagens fornecem quadros de diálogo fáceis de usar.

2.1.12 MATIZ E SATURAÇÃO

Assim como se pode ajustar as definições de televisão, também pode-se ajustar o matiz (*Hue*) ou a cor de qualquer figura em um programa de processamento de imagens. Pode-se melhorar o matiz de uma imagem para deixar as cores mais vívidas ou realistas, freqüentemente. A saturação (*Saturation*) também pode ser controlada através do *software*. A saturação descreve o quanto um *pixel* está saturado de uma cor pura. Uma cor que não contém saturação será uma nuance de cinza; quanto mais saturação se acrescentar, menos cinza existe na cor.

2.1.13 CORREÇÃO DE CINZA / COR

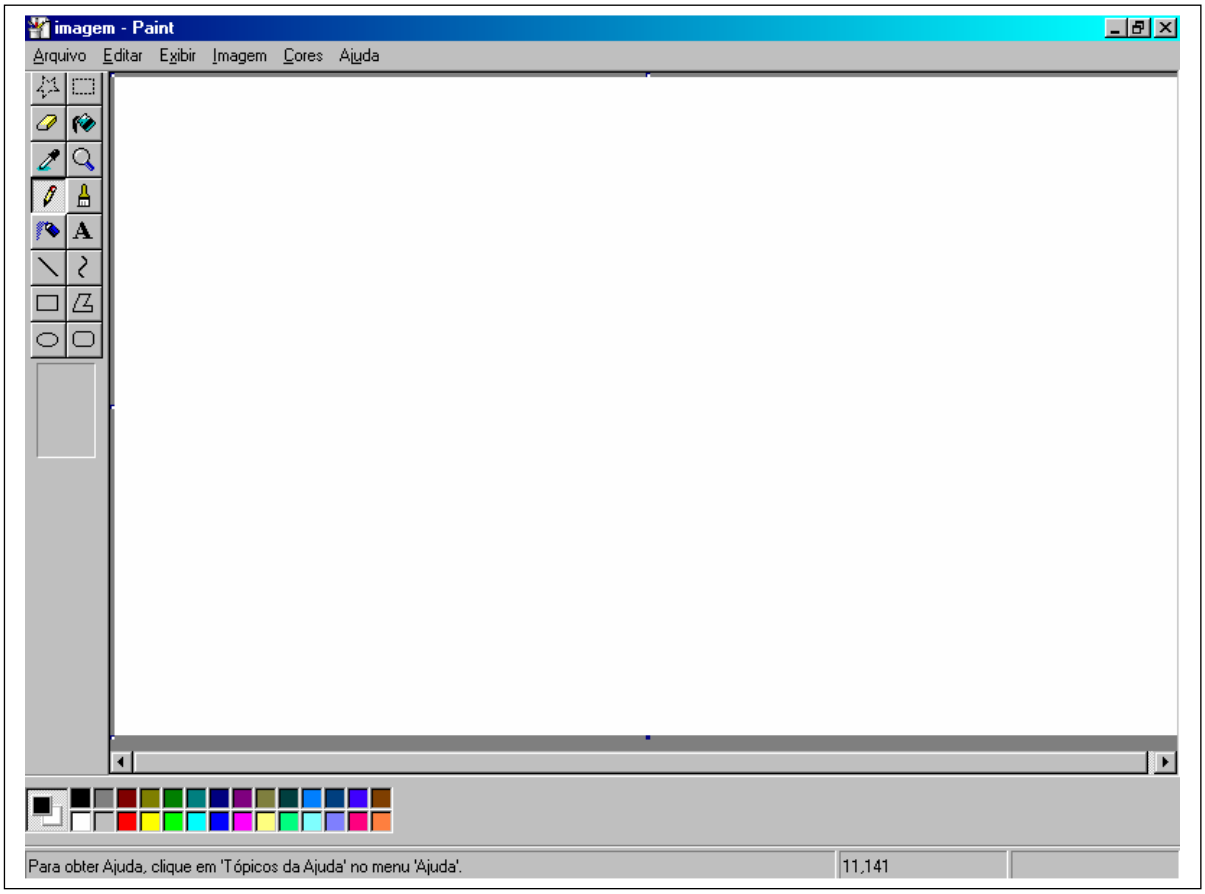
Para obter um controle maior sobre a cor das imagens, existe uma ferramenta chamada de *gray/color Calibration Curve.* Com o uso desta ferramenta é possível executar alterações sem limite nos níveis de brilho de uma imagem. Pode-se definir a curva de calibragem para trabalhar sobre todas as áreas escuras de uma imagem e clareá-las sem afetar as áreas mais claras. Esta ferramenta pode criar o negativo de uma cor deixando as áreas mais escuras o mais claro possível e as áreas mais claras o mais escuro possível ([MOR1995]).

Existem diversos programas que podem ajudar a criar, editar e gerenciar imagens gráficas. O MS-Paint, o Corel Photo-paint, o Jasc Paint Shop Pro e o Adobe Photoshop são programas de visualização e edição de arquivos *raster* 2D que permitem visualizar, editar, converter e imprimir arquivos deste formato.

2.2 PAINT DO WINDOWS 98

O Paint é um aplicativo, fabricado pela Microsoft Corporation, incorporado ao sistema operacional Windows, usado para criação e edição de desenhos no formato *Bitmap*. Como mostra a Figura 1, encontram-se seis opções em seu menu principal, uma pequena barra de ferramentas localizada no lado esquerdo da janela e uma paleta de cores localizada na parte inferior da janela.

Figura 1 – Tela Inicial do Paint do Windows 98



A opção Arquivo deve ser utilizada quando deseja-se executar operações relacionadas a arquivos como Abrir, Salvar e Visualizar impressão. O Paint apenas abre e salva arquivos no formato Bitmap (BMP).

O Paint permite copiar, recortar, e colar objetos no arquivo. As ferramentas de seleção oferecidas são da figura geométrica do retângulo ou feitas a mão livre. Possui algumas

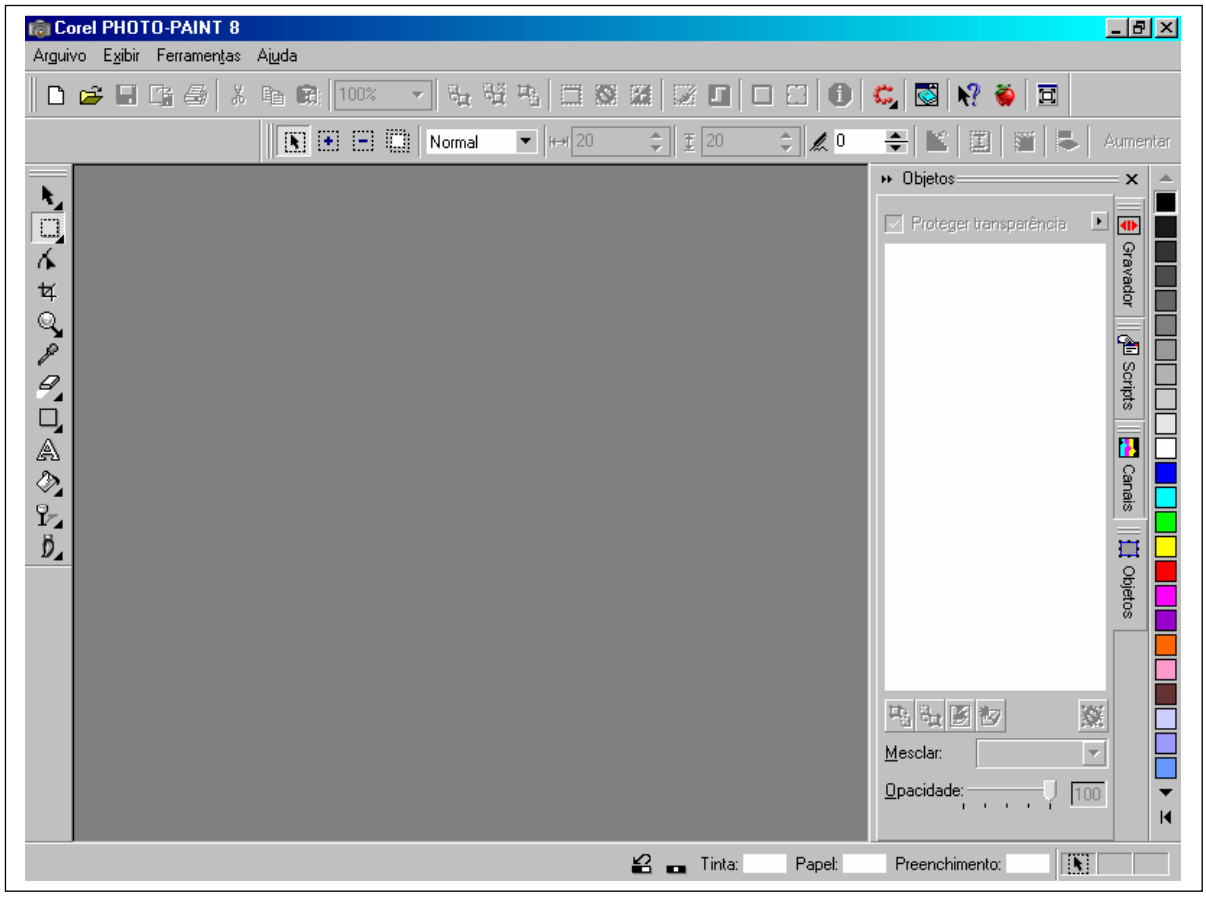
funções onde pode-se criar alguns efeitos, tais como inverter, girar, alongar, inclinar, inverter cores além da possibilidade de desenhar com a definição opaco[1].

Para edição da imagem o Paint oferece algumas opções para desenhar (retas, *splines*, cículos, retângulos e a mão livre), escrever textos, apagar, pintar regiões, pintar usando spray[2] e selecionar cor ([GOL1996]).

2.3 COREL PHOTO-PAINT

O Corel Photo-Paint é parte integrante do pacote do Corel Draw fabricado pela Corel Corporation. A tela inicial do Corel Photo-Paint é apresentada na Figura 2 que apresenta quatro opções em seu menu principal e, ao longo da janela, possui várias opções para facilitar seu uso ([COR1994]).

Figura 2 – Tela Inicial do Photo Paint 8



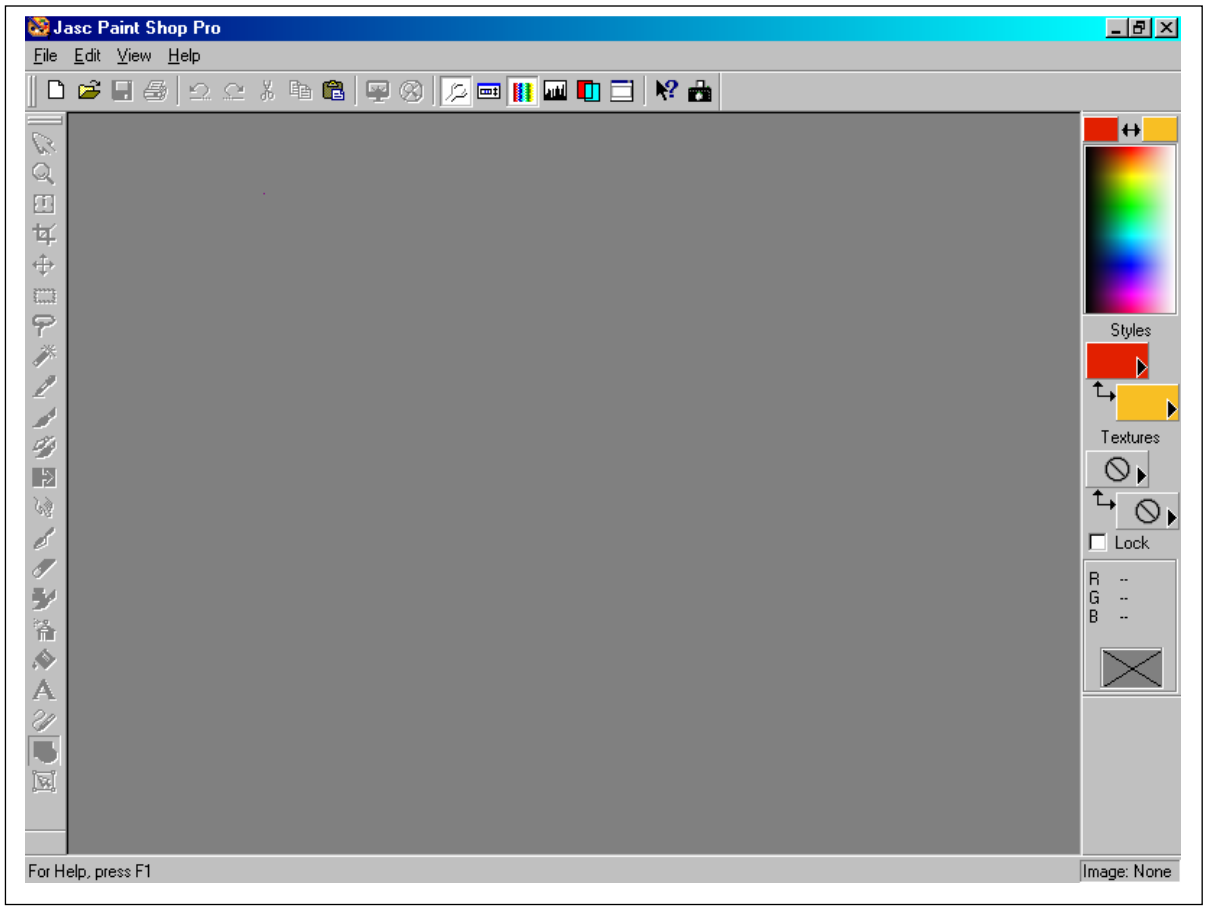
[1] Menor nível de brilho.
[2] Borrifar com a cor de primeiro plano em torno de um ponto selecionado pelo usuário.

O Photo-Paint além do seu próprio formato (CPT), trata diversos outros, inclusive abre figuras 3D (3DMF), de animação (MPG) e documentos (PDF), tratando ao todo mais de 40 formatos distintos.

2.4 JASC PAINT SHOP PRO

O Jasc Paint shop Pro foi desenvolvido pela Jasc *Software* para editar imagens gráficas no padrão *raster* 2D. Na Figura 3 é mostrada a sua tela inicial onde existem quatro opções no menu principal, uma barra de ferramentas localizada no lado esquerdo da janela e ao lado direito opções de textura e cor.

Figura 3 – Tela Inicial do Jasc Paint Shop Pro 7



O Jasc Paint Shop Pro permite abrir e salva arquivos no formato *raster* 2D, oferece também ferramentas de seleção como laço, e varinha mágica além das técnicas tradicionais de manipulação de imagens como: copiar, colar, inverter, espelhar, girar, recortar e

redimensionar. Este programa destaca-se pelas técnicas de filtragem de imagem e pela manipulação de cor que oferece.

Algumas destas técnicas são de realce e contorno de bordas, borrar, aumento de nitidez e suavização entre *pixels* vizinhos e realce de cores.

Pode também transformar a imagem colorida em imagem com escala de cinza (*grayscale*), inverter a imagem para criar um negativo da original, ajusta as matrizes de vermelho, verde e azul (padrão RGB) da imagem. Além destes o Jasc Paint Shop Pro possui diversos filtros que também tratam a cor da figura como: o *Dilate* que realça as cores mais brilhantes da imagem; o *Median* que tira a média das cores, do brilho e do contraste*; Count Colors Used* que conta o número de cores utilizado na imagem; entre outras.

O Jasc Photo Paint Pro não contém nenhuma ferramenta de mapa de *bits* em estilo pintura para desenhar *pixels* individuais no mapa. O programa é projetado para editar imagens de mapa de *bits*, não para criar integralmente desenhos de mapa de *bits* ([COR1994]).

Esta ferramenta trata mais de 40 formatos além dos seus próprios (PSD, JSL, PFR e TUB), entre eles arquivos no formato *metafile* (CGM), arquivos JPEG (JPG), arquivos de Photo CD (PCD), e do Corel Draw (CDR - Corel Draw Drawing).

2.5 ADOBE PHOTOSHOP

O Photoshop é um programa desenvolvido pela Adobe Systems, Inc. para tratar figuras *raster*. Ao iniciar o Photoshop (Figura 4), existe um menu principal com nove opções. No lado esquerdo se apresenta a caixa de ferramentas que contém botões para as funções mais comuns do Photoshop. No lado direito existem janelas, chamadas de paletas, que ajudam no trabalho de edição de imagens. Na parte inferior da tela existe a barra de *status* que fornece informações sobre a imagem e a ferramenta com a qual se está trabalhando ([MOR1995]).

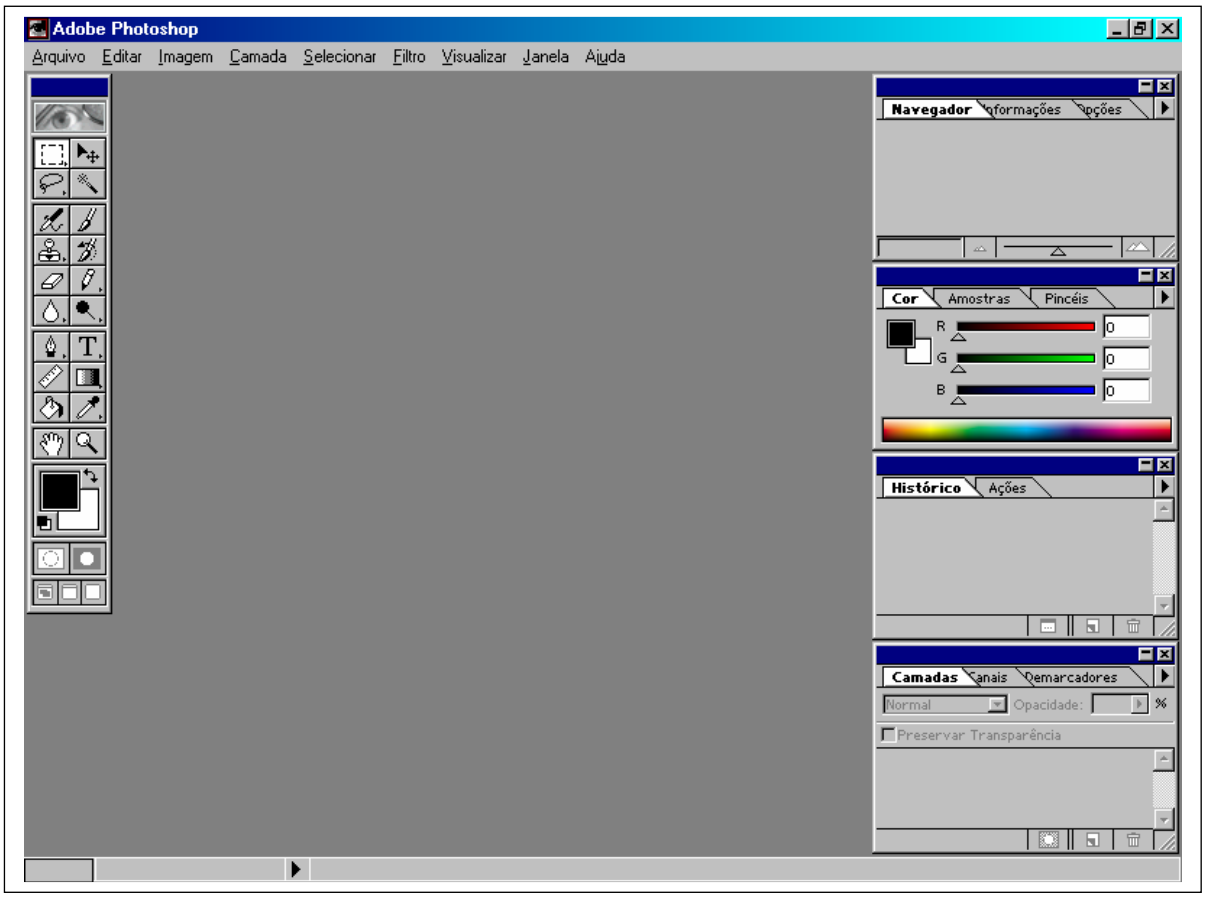
O Adobe Photoshop também oferece as opções básicas de abertura e gravação de imagens gráficas para arquivos *raster* 2D além de variedade de formas para corrigir e ajustar o brilho e o contraste, o balanço de cores, o matiz e a saturação de uma imagem, e também possui as ferramentas principais para seleção como: laço, varinha mágica e caneta.

Para operações que se deseja executar para edição o Photoshop apresenta uma ampla variedade de ferramentas comuns para escolha de cores padrões de primeiro e segundo plano, definir largura de linhas, escrever textos, entre outras.

Para uso mais avançado o Photoshop apresenta ferramentas como carimbo (*rubber Stamp*), *smudge*, alguns filtros de suavização, e de efeitos especiais.

Os filtros de efeitos especiais podem criar: ilusão de movimento dentro da imagem (filtro *Motion Blur*); aparência de como a imagem tivesse sido enrolada em volta de uma esfera (filtro *Spherize*); simulação de ondas (filtro *Wave*); entre outros ([MOR1995]).

Figura 4 – Tela Inicial do Adobe Photoshop 5



O Photoshop, além do seu próprio (arquivos PSD), trata mais de 20 formatos de arquivos.

Segundo McClelland ([MCC1999]), o programa gráfico Adobe Photoshop é o aplicativo para edição de imagens mais utilizado em computadores *Macintosh* e *Windows*,

atualmente. Apesar da concorrência acirrada entre os programas, a Adobe Systems, Inc. ([ADO2000]) informa que o Photoshop detém mais de 80% das vendas no mercado de edição de imagens. Isso torna o Photoshop quatro vezes mais usado do que todos os concorrentes juntos.

A vantagem de vendas sempre superior do Photoshop vem sendo para a Adobe um incentivo para reinvestir no programa e regularmente aperfeiçoar – e até, em alguns casos, corrigir – seus recursos. Embora os concorrentes possam oferecer recursos interessantes, o somatório de suas partes ainda permanece inferior ao Photoshop ([MCC1999]).

Na Tabela 1 é apresentada uma relação entre estas ferramentas com as características discutidas na seção 2.1. É importante destacar que o protótipo pretende implementar apenas algumas técnicas e que as demais podem ser implementadas em novos trabalhos.

Tabela 1 – Relação de Software e suas Características

Característica	Paint	Corel	Jasc	Adobe	Protótipo
Corte e colagem	Sim	Sim	Sim	Sim	Não
Destaque	Sim	Sim	Sim	Sim	Não
Seleção de formas geométricas	Sim	Sim	Sim	Sim	Não
Seleção utilizando ferramenta laço	Não	Sim	Sim	sim	Não
Seleção utilizando ferramenta caneta	Não	Sim	Sim	Sim	Não
Seleção utilizando ferramenta varinha mágica	Não	Sim	Sim	Sim	Não
Ferramenta apagar	Sim	Sim	Sim	Sim	Não
Ferramenta borrar	Sim	Sim	Sim	Sim	Não
Ferramenta clone	Não	Sim	Não	Sim	Não
Ferramenta de embaçamento e nitidez	Não	Sim	Não	Sim	Não
Ferramenta de correção de brilho e contraste	Não	Sim	Não	Sim	Não
Ferramenta de correção na matiz e saturação	Não	Sim	Não	Sim	Não
Ferramenta de correção de cinza e cor	Não	Sim	Sim	Sim	Sim
Ferramenta de rotação	Sim	Sim	Sim	Sim	Sim
Ferramenta de preenchimento de regiões	Sim	Sim	Sim	Sim	Sim
Ferramenta de espelho / reflexo	Sim	Sim	Sim	Sim	Sim

3 ASPECTOS DE ARQUIVOS GRÁFICOS E TRATAMENTO DE CORES

Este capítulo tem o propósito de abordar alguns conceitos relacionados a computação gráfica, com o objetivo de esclarecer alguns elementos que são utilizados em processamento de imagens digitais.

A computação gráfica é totalmente voltada a gerar, processar (tratar) ou interpretar informações visuais. Estas informações estão contidas dentro de uma imagem. Imagem é a representação gráfica, plástica ou fotográfica de pessoa ou de objeto ([FER1986]). Em computação gráfica pode-se definir imagem como sendo um plano onde cada ponto possui um valor representativo da sua intensidade luminosa ([PAZ1988]).

Para uma imagem se apresentar de uma forma que o computador consiga entendê-la e conseqüentemente reproduzir sua visualização, deve ser feita a descrição de seus objetos tais como segmentos de reta, polígonos, esferas, etc. A este procedimento dá-se o nome de síntese de imagem.

Portanto síntese de imagens é uma sub-área da computação gráfica que se ocupa da produção de representações visuais a partir de especificações geométrica e visual de seus componentes ([PER1989]).

Existem vários dispositivos, associados à várias aplicações que captam imagens de objetos e que posteriormente podem ser armazenadas em meios computacionais. Como exemplo tem-se o radar, usados na obtenção de imagens de objetos distantes: aparelhos médicos baseados em ultra-som usados para analisar o interior do corpo humano, a TV na obtenção de imagens locais, e o *scanner* usado para captar figuras. Basicamente o processo de formação da imagem nestes diversos dispositivos é o mesmo ([PAZ1988]).

O dispositivo mais comum para inserir imagens gráficas em um computador é o *scanner*. Os *scanners* são semelhantes às máquinas copiadoras, exceto que eles armazenam a imagem eletronicamente em vez de transferí-la para outro pedaço de papel ([MOR1995]).

Muitos *scanners* expõem uma imagem a uma luz brilhante. Depois, receptores eletrônicos chamados *chrarged-coupled device* (CCD – dispositivo de carga acoplada)

funcionam como sensores para capturar essa luz e convertê-la em pulsos eletrônicos. Esses pulsos são traduzidos para números . Os *scanners* podem ser preto-e-brancos ou coloridos. A resolução dos *scanners* varia de 100 pontos por polegada (dpi) até 2.000 dpi e acima ([MOR1995]).

Após a captação de uma imagem do mundo real, para se manter seu registro permanente, surge a necessidade de armazenamento em algum dispositivo computacional. A tecnologia dos arquivos gráficos foi criada para que armazenem estas imagens.

3.1 ARQUIVOS GRÁFICOS

Segundo Levine ([LEV1993]) um arquivo gráfico é nada mais que um arquivo que armazena uma figura.

Atualmente existe uma grande variedade de formatos de arquivos de imagens, devido principalmente aos avanços dos gráficos por computador ao longo dos anos. À medida que os gráficos por computador amadureceram, resultando em resoluções mais altas e mais cores, novos padrões tiveram de ser criados.

Quando o campo da computação gráfica estava nos seus primórdios as imagens eram, apenas preto-e-branco e seu armazenamento era muito fácil. Formatos comuns foram criados para organizar os dados da imagem em arquivos. Entretanto, quando as imagens com escala de cinza e coloridas passaram a existir, os formatos projetados anteriormente não funcionavam mais com elas. Assim, outro método foi desenvolvido ([MOR1995]).

Para representar imagens coloridas existem algumas formas diferentes de fazê-lo em dispositivos computacionais. O mais comum existente em imagens de mapa de *bits* é o RGB (*Red, Green, Blue*), usado por quase todas as controladoras de vídeo.

No modelo RGB cada *pixel* é composto por três componentes que representam numericamente a quantia de vermelho, verde e azul (*Red*, G*reen and Blue*) para o *pixel*. Para cada componente, o valor zero representa ausência de cor e o máximo valor é toda a quantia de cor. Se existem oito *bits* por componente (0, 0, 0) representa preto, (255, 0, 0) representa o vermelho (*bright red*), (255, 255, 255) representa o branco e, na maioria dos formatos, cada *pixel* será armazenado com 24 *bits* ([LEV1993]).

A quantidade de *bits* que é utilizada para armazenar o valor numérico que representa a cor é chamada de profundidade de *bits* ou *depth*. Quanto maior a quantidade de cores que a imagem possui maior o número de *bits* necessários para armazenar suas cores ([MOR1995]).

Nas imagens preto-e-branco, cada *pixel* pode ter somente um entre dois valores: preto ou branco. Portanto um único *bit* pode armazenar um *pixel* inteiro. Para as imagens que exigem quatro cores, pode-se ter somente dois *bits* para armazenar as informações para cada *pixel*. Dobrando o número de *bits* para quatro, consegue-se 16 cores por *pixel*. Dobrando novamente para 8 *bits*, consegue-se 256 cores possíveis por *pixel*. Dobrando novamente, chega-se a 16 *bits* ou seja 65.536 cores possíveis. E finalmente, com 24 *bits* atinge-se 16.777.216 possibilidades de cores distintas ([MOR1995]).

Ao longo de um certo período de tempo, à medida que os gráficos coloridos foram desenvolvidos, mais métodos ou formatos foram criados para armazenar essas imagens.

Outra questão que afeta o formato dos arquivos de imagens é a compressão. Como as imagens ocupam muito espaço de armazenamento os programadores desenvolveram métodos de compressão de informações. Algumas técnicas de compressão de imagens vieram de outros campos, como por exemplo, a compressão de textos. Como as figuras costumam utilizar grandes espaços em memória ou disco, a compressão se tornou importante.

Usando técnicas de compressão, pode-se conseguir diminuir drasticamente o tamanho de arquivos de imagens. Por exemplo, uma imagem descomprimida que requer 921.000 bytes de armazenamento, precisa de apenas 100.000 bytes de armazenamento depois da compressão. Isso chega a um fator de compressão de noventa porcento, ou 9 por 1 (9:1), sendo que o original pode ser reconstituído de forma idêntica a partir da versão comprimida ([MOR1995]).

Alguns formatos atingem níveis mais altos de compressão, em torno da faixa de 22:1. Isso faria com que um arquivo de 921.000 bytes ficasse com apenas 42.000 bytes. As taxas de compressão desse nível geralmente exigem que dados desnecessários sejam jogados fora, simplificando assim a imagem e reduzindo o seu tamanho. Esses métodos de compressão são chamados de *lossy* porque eles perdem alguns dos dados durante o processo de compressão ([MOR1995]).

Os arquivos gráficos para armazenas figuras de duas dimensões (2D), podem ser do tipo *raster* ou vetorial. Sendo que o primeiro armazena cada *pixel* da imagem em formato seqüencial e o segundo armazena as equações matemáticas que descrevem a geometria da imagem.

A desvantagem dos arquivos vetoriais é que eles não representam muito bem as imagens de tons contínuos. As imagens de tons contínuos contêm tons em escala entre preto até branco ou de uma cor para outra. Portanto, eles são adequados para desenhos de caracteres e ilustrações técnicas. Os formatos *raster*, por sua vez, funcionam bem para as imagens de tons contínuos, mas se aumentadas demais, a figura passa a ser visualizada como se fosse formada por um conjunto de pequenos quadrados. E as imagens de vetores podem ser aumentadas várias vezes sem perder a qualidade, pois as linhas são recalculadas e redesenhadas ([MOR1995]).

3.1.1 ARQUIVOS *RASTER*

Segundo Levine ([LEV1993]) um formato *raster* é um vetor de pontos, onde cada um destes pontos é chamado *pixel*. Cada *pixel* é armazenado com um número que representa sua cor.

Segundo Gonzalez ([GON1992]) um arquivo *raster* pode ser considerado como sendo uma matriz cujos índices de linhas e de colunas identificam um ponto na imagem, e o correspondente valor do elemento da matriz identifica a cor naquele ponto. Os elementos dessa matriz digital são chamados de elementos da imagem, elementos da figura, célula, *pixel* ou *pels*, estes dois últimos, abreviações de *picture elements* (elementos de figura).

A seguir (Tabela 2) encontra-se formatos comuns utilizados para arquivos *raster* de duas dimensões ([MOR1995]).

Tabela 2 – Alguns Formatos para Arquivos *Raster*

Formato	Nome	Fabricante	Cor	Compressão
BMP	Bitmap	Microsoft Windows Corporation	1 a 24 *bits*	Sim
FIF	Fractal Image Format	Interated Systems	8 a 24 *bits*	Sim
GIF	Graphics Interchange Format	CompuServe Incorporated	1 a 24 *bits*	Sim
JPEG	Join Photographics Experts Group	ANSI	8 a 24 *bits*	Sim
PCD	Photo CD	Eastman Kodak Corporation	24 *bits*	Sim
PCX	PC Paintbrush	ZSoft Corporation	1 a 24 *bits*	Sim
PSD	Adobe Photoshop	Adobe Systems Inc	8 a 24 *bits*	Sim
TGA	Targa	Truevision Incorporated	16 a 24 *bits*	Sim
TIFF	Tagged Image File Format	Aldus Corporation	1 a 24 *bits*	Sim

3.2 TRATAMENTO DE CORES

Cor é um aspecto da visão; consiste em uma resposta psíco-física da reação do olho e uma resposta interpretativa automática do cérebro, da característica do tamanho da onda da luz sobre um certo nível de brilho (para níveis baixos, os olhos percebem brilhos diferentes, mas são incapazes de fazer discriminação de cores) ([GON1992]).

Aquela luz de origem da cor foi primeiro demonstrada em 1666 por Isaac Newton, que passou um raio de luz solar através de um prisma de vidro, produzindo um arco-íris de coloração, de um espectro visível. Este fenômeno foi observado antes, mas sempre eram relatadas cores latentes que existiam no vidro do prisma. Newton todavia, tomou um simples experimento um passo adicional. Ele passou seu arco-íris miniatura através de um segundo prisma que reconstituiu o original raio de luz branco. Sua conclusão foi revolucionária: cores estão na luz, não no vidro, e a luz que as pessoas vêem tão brancas são uma mistura de todas as cores de espectro visível.

A razão dos arco-íris aparecerem coloridos é porque a luz é separada dentro das partes constituintes, passando através dos pingos de água no ar.

O uso da cor em computação gráfica apresenta várias vantagens: torna os terminais de vídeos mais bonitos e agradáveis; auxilia a visualização de conexões em desenhos complexos, melhorando a legibilidade da informação; possibilita gerar imagens realistas e permite indicar

mecanismos de segurança (por exemplo, a cor vermelha pode indicar necessidade de atenção para situações críticas). Enfim, o uso de cores torna o processo de comunicação mais eficiente ([BRO1995]).

A cor, elemento fundamental em qualquer processo de comunicação, merece uma atenção especial. É um componente com grande influência no dia a dia de uma pessoa, interferindo nos sentidos, emoções e intelecto ([MAR1987]). Pode, portanto, ser usada deliberadamente para se atingir objetivos específicos. Um projetista de interface pode utilizar do poder das cores para criar interfaces mais poderosas.

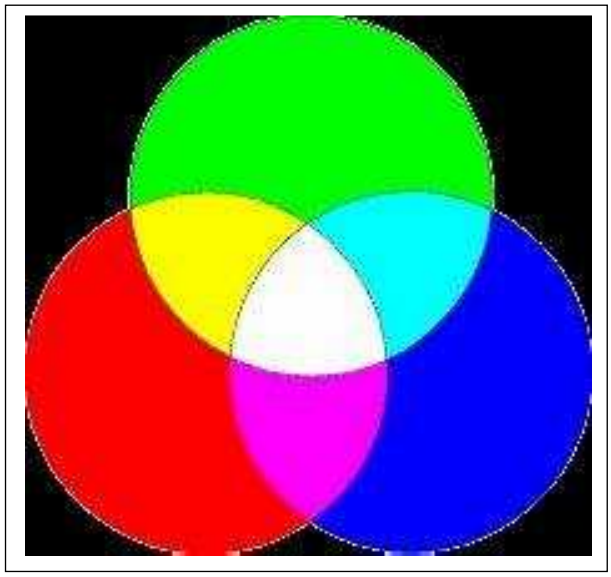
A cor exerce uma função tríplice: a de impressionar, a de expressar e a de construir. A cor é vista: impressiona a retina. É sentida: provoca uma emoção. É construtiva, pois tendo um significado próprio, possui valor e símbolo, podendo assim, construir uma linguagem que comunique uma idéia ([FAR1987]).

O conceito de cor é extremamente importante e indispensável não só para a computação gráfica, pois com o seu uso pode-se explorar um dos principais sentidos, a visão. Os seres humanos podem detectar cerca de 100 níveis de intensidade de luz. O desafio do sistema digital é reproduzir esta resposta convertendo a informação recebida em um número apropriado de cores. Sabendo-se que todas as cores que o olho humano pode perceber são combinações de três cores primárias: vermelho, verde e azul (*red*, *green* e *blue*) ([BRO1995]).

Existem dois processos de produção de cores: mistura aditiva de cores e mistura subtrativa de cores. Esses métodos usam cores primárias diferentes e possuem significados distintos para o branco e para o preto ([RUS1995]).

O processo de mistura aditiva de cores (Figura 5) é usado nos monitores de vídeo e televisões, onde, a cor é gerada através da mistura de vários comprimentos de onda de luz; isso provoca uma alteração do comprimento de onda que atinge e sensibiliza o olho.

Figura 5 – Processo Aditivo de Formação de Cores



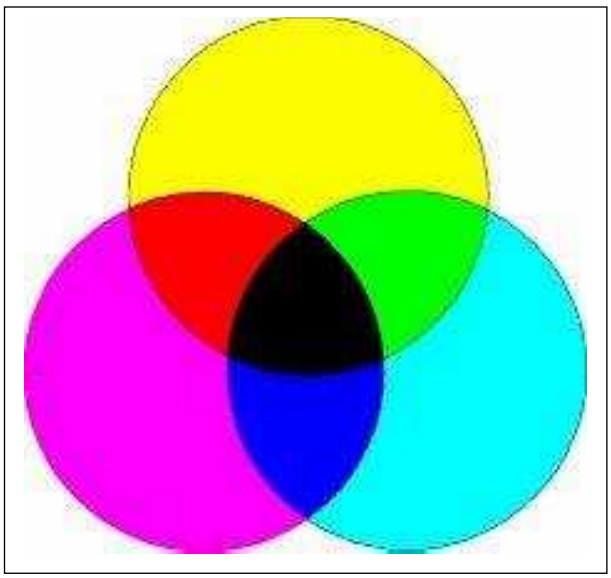
As cores primárias aditivas são o vermelho, verde e o azul. No processo aditivo, o preto é gerado pela ausência de qualquer cor, indicando que nenhuma luz está sendo transmitida; o branco é a mistura de todas as cores o que indica que uma quantidade máxima de vermelho, verde e azul está sendo transmitida. O outro processo de formação das cores é o de mistura subtrativa de cores ([RUS1995]).

O processo de mistura subtrativa de cores é usado por pintores. Uma pintura é diferente de um monitor que, por ser uma fonte de luz, pode criar cores. Uma pintura não emite luz; ela absorve e reflete a luz e, portanto, geram cor através de um processo que absorve comprimentos de ondas de luz específicos e reflete outros.

As cores primárias subtrativas são: *magenta*, amarelo e *cyan*; são cores primárias subtrativas (Figura 6), pois seu efeito é subtrair, isto é, absorver alguma cor da luz branca. Quando a luz branca passa por um objeto, ela é parcialmente absorvida pelo objeto. À parte que não é absorvida é transmitida, determinando assim a cor do objeto ([RUS1995]).

O processo subtrativo altera a cor através de uma diminuição dos comprimentos que são absorvidos. O branco é gerado pela ausência de qualquer cor e o preto é a presença de todas ([RUS1995]).

Figura 6 – Processo Subtrativo de Formação de Cores



Como mostra a Figura 6 o *cyan* absorve a componente vermelha da luz branca refletida (a luz branca é a soma das cores azul, verde e vermelho) assim, como pode ser visto na Tabela 3, em termos de cores subtrativas, *cyan* é a soma de verde e azul; *magenta* retira a componente verde da luz branca, sendo, assim, a soma de vermelho e azul; e o amarelo subtrai a componente azul da luz branca refletida e é a soma do verde e o vermelho ([BRO1995]).

Tabela 3 – Relação Entre Processos de Cores Primárias

Subtrativo	Aditivo
Cyan	Verde + azul
Magenta	Vermelho + azul
Amarelo	Vermelho + verde
Branco	Vermelho + verde + azul
Vermelho	Branco - verde - azul
Verde	Branco - vermelho - azul
Azul	Branco - vermelho - verde

3.2.1 ESCALA DE CINZA

Uma reprodução boa de uma imagem requer um balanço entre claro e escuro. Existe um contínuo de sombras de cinzas entre a cor branca, de uma superfície idealmente refletida, e a cor preta, de uma superfície absorvente, que chama-se escala de cinza. Cores em escala de cinza podem ser descritas por um único valor que representa a intensidade da luz ([BRO1995]).

O número de escalas de cinza perceptíveis depende da sensibilidade e do estado de adaptação do olho, mas geralmente é em torno de cem ([JAC1994]).

4 TÉCNICAS DE PROCESSAMENTO DE IMAGENS

Outra sub-área da computação gráfica é a de processamento de imagens. Após a captação, as imagens são então transformadas em outras com uma representação visualizável onde características visuais são alteradas ([PER1989]).

O interesse em métodos de processamento de imagens digitais decorre de duas áreas principais de aplicação: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para percepção automática através de máquinas. Uma das primeiras aplicações de técnicas de processamento de imagens da primeira categoria foi o melhoramento de imagens digitalizadas para jornais, enviadas por meio de cabo submarino de Londres para Nova York. A introdução do sistema Bartlane de transmissão de imagens via cabo no início dos anos 20 era capaz de codificar imagens em cinco níveis de brilho distintos. Essa capacidade foi aumentada para quinze em 1929 ([GON1992]).

Melhoramento nos métodos de processamento para transmissão de figuras digitais continuaram a ser feitos ao longo dos 35 anos que se seguiram. Entretanto, foi necessária a combinação do surgimento de computadores digitais de grande porte com o programa espacial para que o potencial dos conceitos relativos a processamento de imagens começassem a ser reconhecidos. O emprego de técnicas de computação para o melhoramento de imagens produzidas por uma sonda espacial iniciou-se em 1964, quando imagens transmitidas da Lua foram processadas por um computador para corrigir vários tipos de distorção da imagem inerentes à câmera de televisão a bordo. Essas técnicas serviram de base para métodos melhorados no realce e restauração de imagens de outras missões espaciais subseqüentes ([GON1992]).

De 1964 até hoje, a área de processamento de imagens vem crescendo vigorosamente. Além de aplicações no programa espacial, técnicas de processamento de imagens digitais são atualmente utilizadas para resolver uma variedade de problemas. Embora freqüentemente não relacionados, esses problemas comumente requerem métodos capazes de melhorar a informação visual para a análise e interpretação humanas ([GON1992]).

Em medicina, por exemplo, procedimentos computacionais melhoram o contraste ou codificam os níveis de intensidade em cores, de modo a facilitar a interpretação de imagens de

raios X e outras imagens biomédicas. Geógrafos usam técnicas idênticas ou similares para estudar padrões de poluição em imagens aéreas ou de satélites. Procedimentos para realce e restauração de imagens são usados para processar imagens de objetos irrecuperáveis ou resultados experimentais muito caros para repetição. Em arqueologia, métodos de processamento de imagens têm restaurado com sucesso figuras fotografadas borradas, que eram os únicos registros disponíveis de artefatos raros perdidos ou danificados. Em física e áreas relacionadas, técnicas computacionais rotineiramente realçam imagens de experimentos em áreas como plasmas de alta energia e microscopia eletrônica. Similarmente, aplicações de processamento de imagens podem ser encontradas em astronomia, biologia, medicina nuclear, apoio da lei, defesa e aplicações industriais.

O processamento de imagens é caracterizado por soluções específicas. Existe portanto uma relação direta entre o tipo de processamento e a aplicação desejada. Deste modo, técnicas que funcionam bem em uma área podem se mostrar totalmente inadequadas em uma outra área. Entretanto um conjunto básico de procedimentos é usado na maioria dos casos ([PAZ1988]).

Após a aquisição da imagem e o seu armazenamento em algum dispositivo computacional, pode-se efetuar as transformações, capazes de realçar as características relevantes ([PAZ1988]).

As transformações celulares (por *pixel*), prestam-se principalmente a buscar uma melhor distribuição das diversas tonalidades obtendo uma imagem com melhor contraste.

Os procedimentos que atuam em regiões, tem por objetivo minimizar efeitos produzidos por cada *pixel* individualmente, considerando também a influência dos *pixels* vizinhos. Com este tipo de procedimento pode-se eliminar ruídos, suavizar regiões de grande contraste, isolar regiões de mudança de contraste, detectar bordas, linhas horizontais, verticais ou inclinadas.

Os procedimentos que atuam na imagem como um todo, tem por objetivo extrair informações descritivas úteis em reconhecimento de padrões, armazenamento e transmissão ([PAZ1988]).

Neste momento serão abordadas técnicas de processamento de imagens. Existem diversas técnicas e cada uma delas se classificam basicamente em dois grupos: em técnicas que fazem transformações na figura e de realce.

Transformações são operações que se aplicam a um ponto (ou conjunto de pontos) com o objetivo de modificar a figura original através de alguma técnica de transformação do tipo rotação ou escala por exemplo ([MAG1986]).

As transformações geométricas estão nos fundamentos de inúmeras aplicações gráficas. Elas podem aparecer em um simples programa para representar *layouts* de circuitos, até em elaboradas simulações de movimento num sistema incluindo alguma forma de animação.

Deve ser ainda observado que algumas das transformações geométricas (translação, rotações em torno de pontos diferentes da origem etc.) são transformações lineares afins. Transformações lineares afins podem ser consideradas como transformações lineares definidas num espaço com uma dimensão a mais, restritas a um dado hiperplano. Essa maneira de ver as coisas é usada em programas gráficos com o sentido de reutilização de rotinas já implementadas, especificamente, o produto de matrizes, para expressar algebricamente composições de transformações geométricas de diferentes tipos ([PER1989]).

As imagens por computador são geradas a partir de uma série de pontos representados por coordenadas destes pontos. Certas mudanças em uma imagem podem ser feitas facilmente executando operações matemáticas nestas coordenadas, isto é, multiplicação de matriz.

A Multiplicação de Matrizes é obtida pela multiplicação de cada elemento das linhas da primeira Matriz com os elementos das colunas da segunda Matriz, e soma-se estas parcelas para obter o elemento da Matriz resultante. Desta forma, a multiplicação só existe se o número de colunas da primeira Matriz for igual ao número de linhas da segunda Matriz. O que resultará numa Matriz com dimensões igual ao número de linhas da primeira Matriz por número de colunas da segunda Matriz, como é ilustrado na Figura 7 a seguir.

Figura 7 – Exemplo de Multiplicação de Matrizes

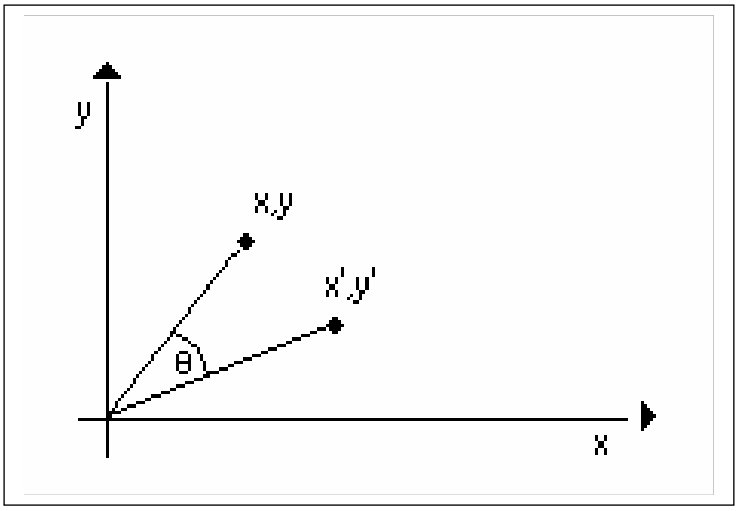
$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}_{(2,3)} \times B=\begin{bmatrix}4 & 1\\ 9 & 3\\ 2 & 7\end{bmatrix}_{(3,2)} \Leftrightarrow C=\begin{bmatrix}((1\times 4)+(2\times 9)+(3\times 2)) & ((1\times 1)+(2\times 3)+(3\times 7))\\ ((4\times 4)+(5\times 9)+(6\times 2)) & ((4\times 1)+(5\times 3)+(6\times 7))\end{bmatrix}_{(2,2)}$$

Agora que viu-se conceitos básicos para transformações geométrica vai-se abordar uma destas técnicas de transformações geométricos: a rotação.

4.1 TÉCNICA DE ROTAÇÃO

Para um ponto (x,y), a rotação em θ (Figura 8) permite obter o ponto (x',y') ([MAG1986]).

Figura 8 – Rotação do Ponto (x, y)



O cálculo do ponto (x',y') é feito conforme a seguir (Figura 9) para rotação no sentido horário ([ROG1990]).

Figura 9 – Fórmula para Rotação Horária

$$[x \quad y]\cdot\begin{bmatrix}\cos\theta & -\operatorname{sen}\theta\\ \operatorname{sen}\theta & \cos\theta\end{bmatrix}=[x' \quad y']$$

Se a rotação fosse no sentido anti-horário o cálculo das coordenadas x’ e y’ seria obedecendo a equação a seguir (Figura 10).

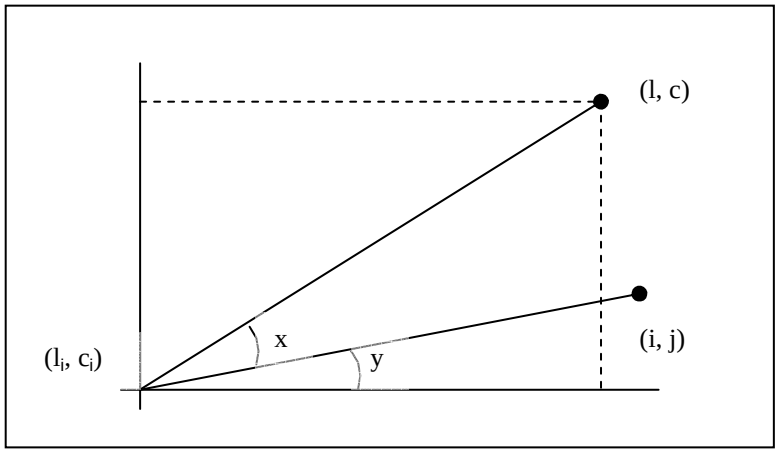
Figura 10 – Equação Matricial para Rotação Anti-horária

$$[x \quad y] \cdot \begin{bmatrix} \cos\theta & \text{sen}\,\theta \\ -\text{sen}\,\theta & \cos\theta \end{bmatrix} = [x' \quad y']$$

Esta equação pode ser utilizada para voltar ao ponto original utilizado na equação da rotação horária ([PAR1993]).

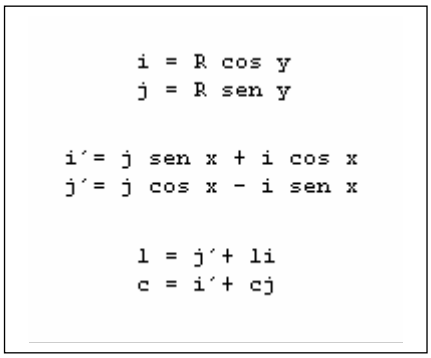
Para imagens *raster* 2D a rotação é feita a partir de uma origem (l_i,c_j), geralmente o ponto central da imagem, onde l_i é o número de linhas total da imagem dividido por dois e c_j é o número total de colunas da imagem dividido por dois. A Figura 11 ilustra as variáveis necessárias para a equação. ([PAR1993]).

Figura 11 – Rotação do *Pixel* para Figuras *Raster*



A coordenada (i, j) representa o ponto que se quer rotacionar e (l, c) o ponto já rotacionado. O grau x é a rotação desejado e y é o grau existente entre o eixo horizontal e a reta formada entre o ponto central e o ponto a ser rotacionado. Originando as equações ilustradas na Figura 12.

Figura 12 – Equações de Rotação para Figuras *Raster*



Onde tem-se:

- *R* como a distância entre, o ponto de rotação e o de origem;
- *i'* e *j'* são pontos intermediários do cálculo.

Um outro enfoque para processamento de imagens é realçar a imagem. O objetivo de realçar uma imagem é gerar uma outra, com melhores características de observação e análise. A imagem resultante de um procedimento de realce pode apresentar uma melhor aparência, ou ser transformada em uma representação onde sejam evidenciadas algumas características ([MAG1986]).

Quando se deseja realçar uma imagem, não tem-se necessariamente o objetivo de manter a fidelidade da imagem inicial, Este objetivo é desejado em restauração de imagens onde procura-se reproduzir a imagem original.

As operações de realce tendem a depender tanto dos dados quanto do observador. Não existe uma teoria geral unificada para realce de imagem pois, não existe um padrão de qualidade de imagem que sirva com meta geral para todas as aplicações.

Normalmente a tarefa de realce é interativa. Sucessivos ensaios serão feitos até que seja determinado um modelo de transformação para obtenção da imagem desejada.

Manchas e iluminação irregular são problemas solucionáveis por realce de imagens ([MAG1986]).

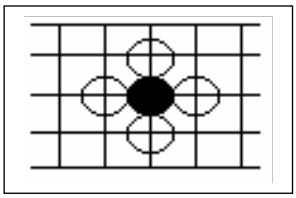
4.2 PREENCHIMENTO DE REGIÕES

Outra técnica de processamento de imagens é a de preenchimento de regiões. Este consiste em coloração do interior de uma dada área ou região ([PLA1991]). Em arquivos *raster*, região é o nome dado a um conjunto conexo de *pixels*. Os *pixels* de uma região poderiam ser caracterizados por uma cor (ou outro atributo), por exemplo ([PER1989]).

Pode-se definir que um conjunto de *pixels* é conexo se, para qualquer par de *pixels* no conjunto, é possível definir-se uma seqüência de *pixels* no conjunto que inicia-se no primeiro e termina no segundo, ou seja, o segundo *pixel* está localizado em uma coordenada imediatamente ao lado, acima e, em alguns casos, numa posição diagonal em relação o primeiro.

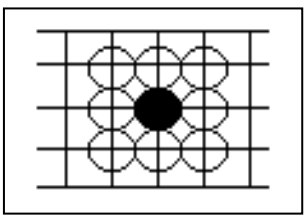
Com esta definição remete-se o problema de conexidade de área ao da conexidade de pares de *pixel*. Há duas alternativas a considerar. Dois *pixels* são 4-conexos se um está imediatamente à esquerda ou acima de outro. Segundo esta definição, os *pixels* do interior da memória da imagem (*pixels* que não fazem parte da primeira nem da última linha, da primeira nem da última coluna) possuem 4 outros a quem são 4-conexos (Figura 13).

Figura 13 – Pixels 4-Conexos



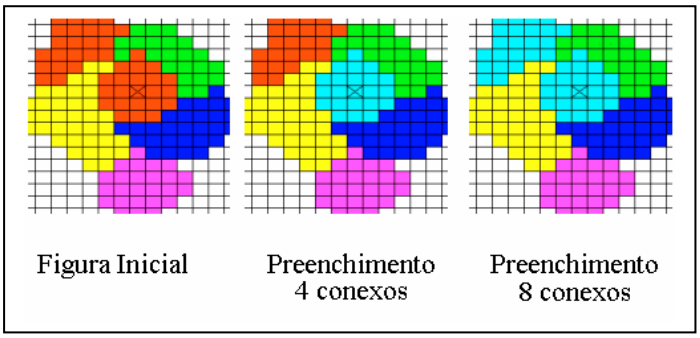
Dois *pixels* são 8-conexos se são 4-conexos ou um está imediatamente acima e à esquerda ou acima e à direita do outro. No interior da memória de imagem cada *pixel* admite 8 *pixels* a que são 8-conexos (Figura 14) ([PER1989]).

Figura 14 – Pixels 8-Conexos



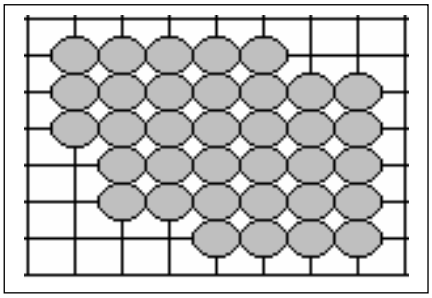
Na Figura 15 são mostrados exemplos de preenchimento de região 4 e 8 conexos.

Figura 15 - Exemplos de Preenchimento 4 e 8 Conexos



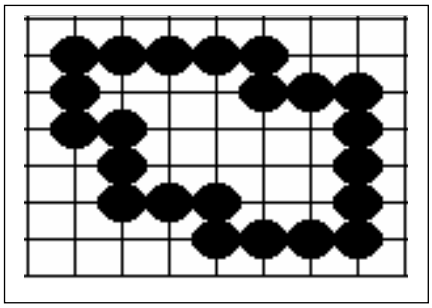
As áreas podem ser descritas ao nível do *pixel* ou geométrico. Ao nível do *pixel*, a área é descrita ou pela totalidade dos *pixels* que a compreendem ou em termos dos *pixels* que a delimitam. No primeiro caso, a área é chamada área definida pelo interior (Figura 16) e os algoritmos usados para o preenchimento de tais áreas são chamados algoritmos de preenchimento por saturação.

Figura 16 – Região Definida de Interior



O outro tipo de área é chamada área definida pela fronteira (Figura 17), sendo os algoritmos associados chamados algoritmos de preenchimento por fronteira ([PLA1991]).

Figura 17 – Região Definida de Fronteira



4.2.1 PREENCHIMENTO POR SATURAÇÃO

Quando é usado o preenchimento por saturação, parte-se de um pixel inicial chamado *gérmen*. A partir do *gérmen*, o algoritmo inspeciona cada um dos pixels conexos para determinar se toda a região já foi coberta. O processo é repetido até que todos os pixels dentro da região tenham sido inspecionados ([PLA1991]).

Para a definição da área de interior faz-se pesquisa nos seus conexos, e, com comparações de cores, pode-se verificar se pertencem a mesma região. Faz-se estas comparações para todos novos *pixels* encontrados, até nenhum novo *pixel* ser achado para a região ([PER1989]).

4.2.2 PREENCHIMENTO POR FRONTEIRA

No preenchimento por fronteira, também existe um *gérmen* (o *pixel* inicial). O algoritmo por fronteira, então, um por um, inspecionará cada *pixel* à direita e à esquerda do *gérmen*. Quando os *pixels* da fronteira, à direita e à esquerda, são atingidos, é desenhada uma linha de *pixels*. A seguir, o algoritmo inspecionará os *pixels* acima e abaixo da linha desenhada. Uma vez mais é desenhada uma linha quando os pixels da fronteira são atingidos. O processo continua até que todos os *pixels* inspecionados sejam *pixels* de fronteira ([PLA1991]).

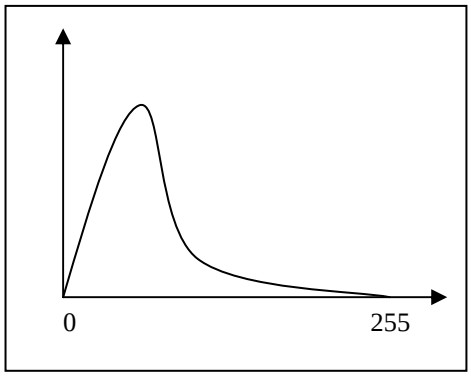
4.3 HISTOGRAMA

As técnicas de realce de imagens tem por objetivo modificar os valores de elemento, em regiões ou na totalidade da imagem. Uma informação importante é a forma da distribuição do número de elementos para cada tonalidade. Histograma de uma imagem é a representação obtida plotando o número de ocorrências de uma cor ou tonalidade de cinza.

Conhecendo-se o histograma, pode-se programar alterações nos valores dos elementos da imagem obtendo uma outra de melhor qualidade ([PAZ1988]).

Para histogramas de imagens em tons de cinza o valor zero representa o preto e o maior valor representa o branco. Na Figura 18 observa-se uma concentração maior de valores próximos do zero, ou seja, a imagem é mais escura ([PAZ1988]).

Figura 18 – Histograma de Uma Imagem em Tons de Cinza



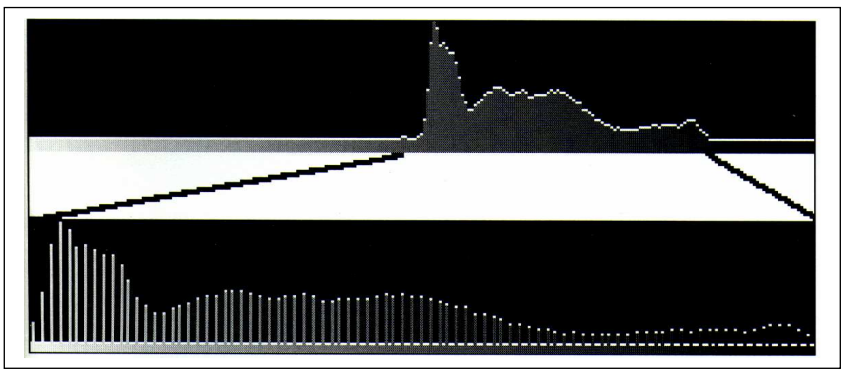
Alguns equipamentos utilizados para captação de imagens, são considerados limitados para serem utilizados com objetivos técnicos e científicos, captando imagens com *depth* (profundidade de cor) menor do que o desejável, gerando imagens com difícil ou quase impossível interpretação pelo olho humano ([RUS1995]).

Verificando-se o histograma destas imagens, pode-se definir uma estratégia de expansão de seu *depht*. Esta técnica é utilizada para melhorar a visualização de imagens captadas por dispositivos que possuem um nível de profundidade de cor e que deseja-se aumentar este nível.

Como uma das aplicações desta técnica pode-se citar as imagens captadas do interior do corpo humano por equipamentos médicos, com a expansão do *depht* da imagem, pequenos detalhes podem ser melhor percebidos pelo olho humano facilitando assim o diagnóstico médico.

Na Figura 19 pode ser visualizado os histogramas de uma imagem onde foi utilizada a técnica de expansão linear ([RUS1995]).

Figura 19 – Histograma Expansão Linear de Imagem



5 DESENVOLVIMENTO DO PROTÓTIPO

Com base nos conceitos apresentados nos capítulos anteriores, tornou-se possível o desenvolvimento do protótipo de uma ferramenta que permite fazer transformações em imagens gráficas de arquivos *Raster* 2D. Neste capítulo, serão abordados a especificação, implementação e o funcionamento do protótipo.

5.1 ESPECIFICAÇÃO DO PROTÓTIPO

Segundo [MEL1990], para o desenvolvimento de sistemas de informação, a prototipação representa uma boa solução para a maioria dos problemas. A metodologia de prototipação de sistemas utilizada neste protótipo é a Prototipação Evolutiva. Conforme [MEL1990], na prototipação evolutiva, o produto final será o próprio sistema, na sua forma mais aperfeiçoada. A prototipação evolutiva é usada na identificação gradual do problema e na construção de modelos concretos, adaptados e corrigidos a medida que o usuário e o analista vão conhecendo a realidade e a solução do problema.

Para o desenvolvimento do protótipo utilizou-se as seguintes fases de especificação:

a) Exame de viabilidade do projeto: onde se verificou os recursos de software e hardware necessários, o planejamento e diretrizes do protótipo;
b) Identificação das necessidades e requisitos do sistema: onde foram identificados e definidos os objetivos do desenvolvimento do protótipo. Nesta etapa utilizou-se técnicas de modelagem de dados como diagrama de contexto e diagrama de fluxo de dados de nível 1;
c) Desenvolvimento do modelo: foi construído o modelo de operacionalização do protótipo;
d) Demonstração e uso do protótipo: nesta etapa foram avaliadas as funções do protótipo;
e) Revisão e melhoramentos: a partir dos problemas identificados na etapa anterior, foram feitas melhorias no programa.

5.1.1 DIAGRAMA DE CONTEXTO

O diagrama de contexto (Figura 20) é uma representação gráfica do sistema como um todo e os seus relacionamentos. Neste diagrama estão representados, em uma visão macro, os

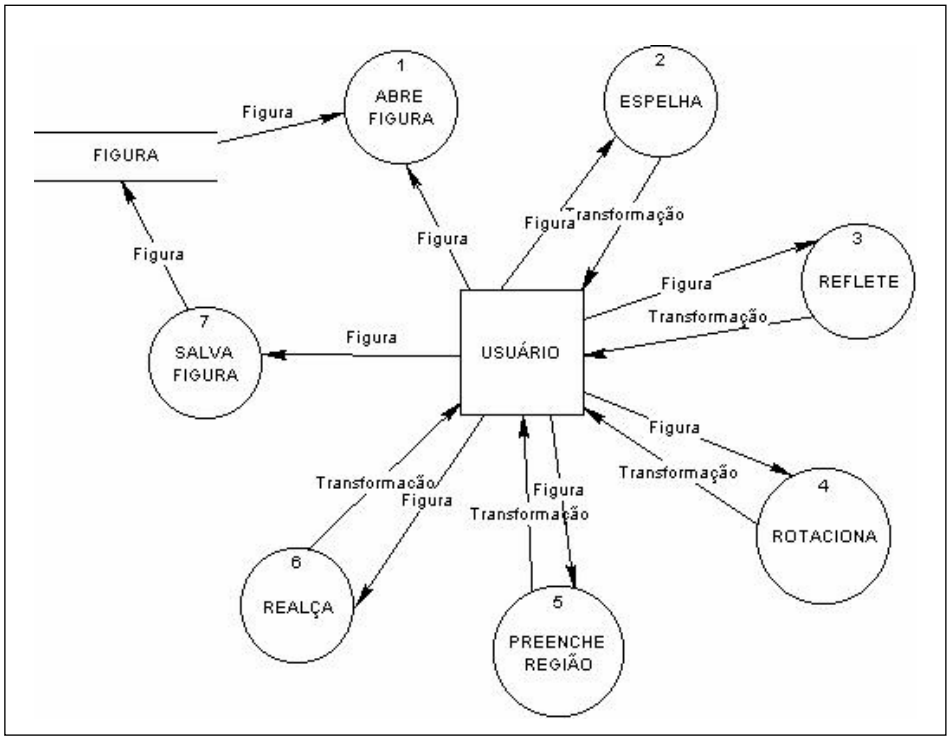
eventos do protótipo bem como o fluxo dos dados. No primeiro evento, o usuário informa ao protótipo qual a figura será aberta e manipulada. No segundo evento o protótipo retorna ao usuário a figura transformada.

Figura 20 – Diagrama de Contexto

5.1.2 DIAGRAMA DE FLUXO DE DADOS NÍVEL 1

Na Figura 21 são melhor detalhadas as funções vinculadas a entidades lógicas de dados.

Figura 21 – Diagrama de Fluxo de Dados Nível 1



O usuário informa ao protótipo qual arquivo armazena as especificações da figura a ser transformada. O protótipo recebe as especificações do arquivo (processo 1) e, dependendo do tipo de processamento solicitado pelo usuário, efetua a sua transformação (processos 2 à 6). O usuário pode, ainda, solicitar a gravação das especificações da figura transformada (processo 7).

5.1.3 DICIONÁRIO DE DADOS

Algumas estruturas são utilizadas para armazenar temporariamente dados, para possibilitar o processamento da imagem.

A estrutura GuardaPos (Tabela 4) é utilizada no processo de rotação. Nela são armazenadas as coordenadas e a cor dos *pixels* que serão utilizados para tratar os ruídos gerados.

Tabela 4 – Estrutura `GuardaPos`

Nome	Tipo	Tamanho (em bytes)	Descrição
CoordenadaX	Inteiro	2	Posição da coordenada X
CoordenadaY	Inteiro	2	Posição da coordenada Y
Cor	Byte	1	Valor da cor

A Tabela 5 descreve a estrutura `TabPontos` que é utilizada no algoritmo de saturação que faz preenchimento de regiões. Esta estrutura armazena os pontos que serão pesquisados no algoritmo.

Tabela 5 – Estrutura `TabPontos`

Nome	Tipo	Tamanho (em bytes)	Descrição
CoordenadaX	Inteiro	2	Posição da coordenada X
CoordenadaY	Inteiro	2	Posição da coordenada Y

A Tabela 6 descreve a estrutura RetaHor. Esta é utilizada no algoritmo de preenchimento de regiões por fronteira para armazenar as coordenadas da reta horizontal pesquisada no algoritmo.

Tabela 6 – Estrutura `RetaHor`

Nome	Tipo	Tamanho (em bytes)	Descrição
CoordenadaX	Inteiro	2	Posição da coordenada X
CoordenadaY	Inteiro	2	Posição da coordenada Y

5.2 IMPLEMENTAÇÃO DO PROTÓTIPO

O protótipo foi desenvolvido para abrir figuras no formato BMP com 24 *bits* de profundidade de cor e teve sua implementação no ambiente *Delphi* 5.0. Este ambiente apresentou componentes importantes para possibilitar o seu desenvolvimento. A propriedade `Tcanvas` do Delphi, é uma classe que contém funções para desenhar figuras geométricas e manipular a grade de *pixels* de uma figura dentro de uma área de desenho de formulários e de outros componentes gráficos como o `Timage`.

O *Delphi* apresenta características de implementação simplificada de janelas do tipo `MDI` (*Multiple Dispositive Interface*) onde formulários filhos podem ser criados. O conceito de formulários `MDI` foi utilizado para abrir uma ou mais figuras e para representar a transformação após o processamento da imagem.

A seguir será feita uma descrição de como as rotinas e técnicas foram implementadas.

5.2.1.1 ESPELHO

Para a implementação desta técnica cada *pixel* da imagem original é lido da esquerda para a direita, iniciando da posição zero e terminando no tamanho máximo de largura (*width*) da imagem, e, em seguida, exibido no formulário de destino, da esquerda para a direita, fazendo com que a imagem pareça como se tivesse sido refletida em um espelho (Quadro 1).

Quadro 1 – Algorítmo de Espelho

```
Var
  Cor:Tcolor;
Begin
  for x:=0 to width-1 do
    for y:=0 to height-1 do
      begin
        Cor:=Figura.Canvas.Pixels[x,y];
        Ffilho.Imagem.Canvas.pixels[width-x,y-1]:=Cor;
      end;
End;
```

5.2.1.2 REFLEXO

Esta técnica consiste em refletir uma imagem onde cada *pixel* da imagem original é lido de cima para baixo, iniciando do *pixel* que corresponde a posição zero até o tamanho total

da de altura (*height*) da imagem, e, em seguida, exibindo no formulário de destino, de baixo para cima, fazendo com que a imagem se torne de cabeça para baixo, causando o efeito de reflexo (Quadro 2).

Quadro 2 – Algorítmo de Reflexo

```
Var Cor:Tcolor;
Begin
  for x:=0 to width-1 do
    for y:=0 to height-1 do
      begin
        cor:=Figura.Canvas.Pixels[x,y];
        Ffilho.Imagem.Canvas.pixels[x,height-y-1]:=Cor;
      end;
End;
```

5.2.1.3 ROTAÇÃO

Ao rotacionar uma figura, as dimensões de altura e largura normalmente são diferentes da figura original. Ao se criar o objeto `TIMAGE,` que hospedará esta figura, as coordenadas de altura (`Height`) e largura (`Width`) devem ser informadas, para tanto, são calculados os quatro cantos da figura transformada e com comparações de menor e maior acham-se os menores e maiores X e Y. Fazendo-se a soma absoluta do menor X com o seu maior e do menor Y com seu maior obtêm-se respectivamente a largura e a altura que a figura transformada vai ocupar. O Cálculo é mostrado no Quadro 3.

Quadro 3 – Algorítmo para Dimensionar a Nova Figura

```
{Localiza os menores e maiores valores para x e y}
for I:=0 to 3 do {índice da estrutura que armazena os quatro cantos}
  begin
    x := (Ponto[I].x * coseno) + (Ponto[I].y * Seno1);
    y := (Ponto[I].x * Seno2) + (Ponto[I].y * coseno);
    if X < menorX then MenorX:=x else
    if X > MaiorX then MaiorX:=X;
    if Y < menorY then MenorY:=Y else
    if Y > MaiorY then MaiorY:=Y;
  end;
{Dimensiona a nova figura}
FFilho.Imagem.Picture.Bitmap.Height:=Round(MaiorY+abs(MenorY))+1;
FFilho.Imagem.Picture.Bitmap.Width:=Round(MaiorX+abs(MenorX))+1;
FFilho.ClientWidth  := FFilho.Imagem.Picture.Bitmap.Width;
FFilho.ClientHeight := FFilho.Imagem.Picture.Bitmap.Height;
```

Após a nova figura ter sido dimensionada, as posições para seus pontos podem ser calculadas e cada pixel pintado. Este cálculo utiliza-se dos conceitos de rotação para figuras geométricas estudados na seção 4.1, conforme descrito no Quadro 4.

Quadro 4 – Algorítmo de Rotação

```
Var
  XOrig, YOrig,                // Coordenadas da figura original
  TotXOrig,                    // Largura total da figura original
  TotYOrig : Integer;          // Altura total da figura original
  XDest, Ydest : Integer;      // Coordenadas da figura destino
  Cor,                         // Cor do pixel que está sendo calculado
  CorFundo : Tcolor;           // Cor de fundo da figura destino
  Grau,                        // Grau de rotação
  Seno1, Seno2,                // Valores de seno utilizados no cálculo
  Coseno : extended;           // Valor de coseno utilizado no cálculo
Begin
 Grau := grau * (3.1415926535 / 180); // transforma graus em radianos
 case GrSentido.ItemIndex of   // opção do usuário do sentido da rotação
    0 : begin                  // sentido Anti-horário
          Seno1 := Sin(grau);
          Seno2 := Sin(Grau)*-1;
        end;
    1 : begin                  // sentido Horário
          Seno1 := Sin(Grau)*-1;
          Seno2 := Sin(Grau);
        end;
  coseno:=cos(Grau)
 for XOrig:=0 to TotXOrig do   // varredura horizontal da imagem original
    for YOrig:=0 to TotYOrig do // varredura vertical da imagem original
      begin
        { capta a cor do pixel da figura original }
        cor:=Figura.Canvas.Pixels[XOrig,YOrig];
        { calcula as  novas coordenadas na figura destino }
        XDest := round(XOrig * coseno + (YOrig * seno1)+MenorX;
        YDest := round(xOrig * seno2 + (yOrig * coseno)+MenorY;
        { se o ponto não está pintado na figura destino}
        if ffilho.imagem.canvas.pixels[XDest,YDest]=CorFundo then
          { atribui a cor do pixel na figura destino }
          ffilho.imagem.canvas.pixels[XDest, YDest]:=Cor
        { se o ponto já está pintado }
        else
          begin {Guarda pontos já pintados na estrutura GUARDAPOS }
            GuardaPos[IGuardaPos].CoordenadaX:=XDest;
            GuardaPos[IGuardaPos].CoordenadaY:=YDest;
            GuardaPos[IGuardaPos].Cor:=cor;
            Inc(IGuardaPos);
          end;
      end;
End;
```

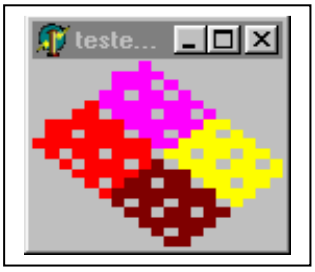
Como conseqüência da técnica de multiplicação de matrizes (utilizada para calcular a rotação) alguns ruídos são gerados para a nova figura calculada, já que os valores das novas

coordenadas podem ser não inteiros necessitando ser arredondadas. Surgindo outra necessidade, o tratamento dos “ruídos” gerados. A Figura 22 apresenta uma imagem original. A partir dela se quer obter uma nova imagem com a rotação de 45 graus no sentido anti-horário.

Figura 22 – Figura Original da Rotação

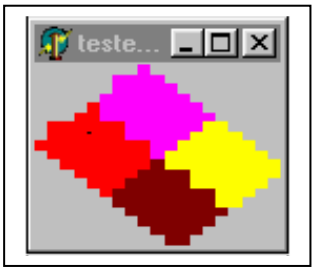
Após a aplicação do algorítmo de rotação obtêm-se a imagem com ruídos ilustrados na Figura 23.

Figura 23 – Figura com Ruídos



Para tratar este problema foram desenvolvidas três técnicas. A primeira técnica implementa para o *pixel* não pintado (x, y), repete o do valor do *pixel* localizado na coluna anterior (x-1, y) obtendo-se o resultado ilustrado na Figura 24.

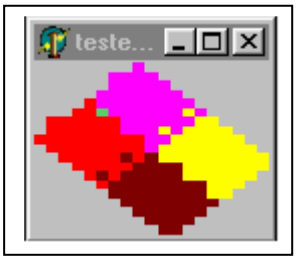
Figura 24 – Primeira Técnica de Correção de Ruídos



A segunda técnica, busca na estrutura `GuardaPos` (descrita em 5.1.3) um ponto cujas coordenadas sejam próximas ao ponto que não foi pintado. Os pontos são armazenados nesta

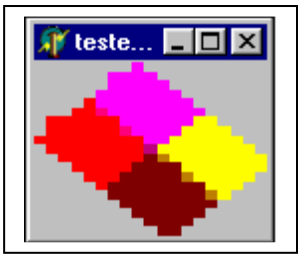
estrutura quando o algoritmo detecta que uma coordenada é repetida. Para esta técnica o resultado obtido é ilustrado na Figura 25.

Figura 25 – Segunda Técnica de Correção de Ruídos



A terceira técnica, faz a média das cores dos *pixels* das colunas anterior e posterior e a atribui ao *pixel* não pintado. Na Figura 26 é ilustrado o resultado obtido com a aplicação.

Figura 26 – Terceira Técnica de Correção de Ruídos



5.2.1.4 PREENCHIMENTO DE REGIÕES

Outro algoritmo desenvolvido é o de preenchimento de regiões, onde, as técnicas descritas na seção 4.2 (saturação e fronteira) são implementadas. Tanto saturação como fronteira partem de um ponto inicial (x, y) onde se inicia o processo de preenchimento.

O ponto inicial da região passa a ser a primeira ocorrência de uma fila que armazenará cada ponto que deve ser verificado, denominada `TabPontos` (seção 5.1.3). O algoritmo faz a varredura dos pontos desta fila verificando os seus quatro ou oito pixels conexos, e se pertencerem a mesma região serão incluídos nesta mesma estrutura. Este processo se repete até o último item da lista ser verificado (Quadro 5).

Quadro 5 – Algorítmo de Preenchimento por Saturação

```
Var

  // Estrutura que guardará os pontos dos pixels a serem verificados
  TabPontos : array [1..100000] of TCoord;

  // Ponteiros de início e final da fila: TabPontos
  I,F : Integer;

// Guarda os pontos pertencentes a região, a serem pesquisados
Procedure GuardaPonto(X,Y:integer);
  var
    IAux : integer;
    begin
      IAux := 0;
      if not (x<0) and not (y<0) then
        repeat  // pesquisa se o ponto já está guardado na estrutura
          inc(IAux);
        until
          ((TabPontos[IAux].x=x) and (TabPontos[IAux].y=y)) or (IAux=F);
        if (TabPontos[IAux].x<>x) or (TabPontos[IAux].y<>y) then
          begin  // se não existir na estrutura, guarda o novo ponto
            inc(F);  // Incrementa a posição final da fila
            TabPontos[F].x:=X;  // Guarda coordenada X (Coluna)
            TabPontos[F].y:=Y;  // Guarda coordenada Y (Linha)
          end;
    end;
Begin

// Inicializa as coordenadas de início e fim da Fila para : TabPontos
 I:=1; F:=1;

// Carrega a primeira posição da fila com as coordenadas do Germen
TabPontos[I].x:=X;   TabPontos[I].y:=Y;

Repeat           // verifica se o ponto pertence a região
 if (FFilho.Imagem.Canvas.Pixels[TabPontos[I].x,TabPontos[I].y]=CorAnt) or
    (FFilho.Imagem.Canvas.Pixels[TabPontos[I].x,TabPontos[I].y]=CorNova)
  then
   begin     // Guarda os ponto 4-conexos ao ponto verificado
    FFilho.Imagem.Canvas.Pixels[TabPontos[I].x,TabPontos[I].y]:=CorNova;
    GuardaPonto(TabPontos[I].x-1,TabPontos[I].y);
    GuardaPonto(TabPontos[I].x,TabPontos[I].y-1);
    GuardaPonto(TabPontos[I].x,TabPontos[I].y+1);
    GuardaPonto(TabPontos[I].x+1,TabPontos[I].y);
    if RGPixels.ItemIndex=1 then   // Se opção for de 8-conexos
     begin        // Guarda os demais pontos para 8-conexos
      GuardaPonto(TabPontos[I].x-1,TabPontos[I].y-1);
      GuardaPonto(TabPontos[I].x-1,TabPontos[I].y+1);
      GuardaPonto(TabPontos[I].x+1,TabPontos[I].y-1);
      GuardaPonto(TabPontos[I].x+1,TabPontos[I].y+1);
     end;
   end;
   inc(I);       // Incrementa variável para outro ponto ser verificado
until (I>F);     // Encerra rotina quando chega ao final da fila
End.
```

A técnica de preenchimento por fronteira parte do ponto inicial e busca horizontalmente (Quadro 6) a fronteira decrementando X, depois faz o mesmo para o outro lado incrementando X. Todos os pontos encontrados e pertencentes a região são pintados e guardados na estrutura de dados `RetaHor` (5.1.3).

Quadro 6 – Fronteira (Linha Horizontal)

```
{traça e guarda linha horizontal da região}
auxX:=x;
auxY:=y;
I:=0;   // Indice para estrutura
Repeat  // repete até encontrar a fronteira do lado esquerdo
 FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]:=CorNova;  // atribui nova cor
 inc(i);
 RetaHor[I].x:=AuxX;   // Guarda a coord. X pertencente a região
 RetaHor[I].y:=AuxY;   // Guarda a coord. Y pertencente a região
 dec(AuxX);
until (FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]<>Corant) or (AuxX<0);
auxX:=x;
auxY:=y;
repeat  // repete até encontrar a fronteira do lado direito
 FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]:=CorNova;  // Atribui nova cor
 inc(i);
 RetaHor[I].x:=AuxX;  // Guarda a coord. X pertencente a região
 RetaHor[I].y:=AuxY;  // Guarda a coord. y pertencente a região
 inc(AuxX);
until (FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]<>Corant);
```

Para cada ponto guardado da linha horizontal é traçada uma linha vertical. Esta linha se inicia do ponto guardado decrementando Y até encontrar a fronteira acima, e em seguida incrementa Y do ponto guardado até encontrar a fronteira abaixo. O algoritmo está descrito no Quadro 7.

Quadro 7 – Fronteira (Linha Vertical)

```
F:=I;
{Traça linhas verticais a partir da linha horizontal traçada anteriormente}
for I:=1 to F do  // pesquisa toda a estrutura RetaHor
  begin
    AuxX:=RetaHor[I].x; AuxY:=RetaHor[I].y;
    Repeat  // repete até encontrar a fronteira vertical superior
      FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]:=CorNova;  // atribui nova cor
      dec(auxY);
    until (FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]<>CorAnt);
    AuxY:=y;
    Repeat  // repete até encontrar a fronteira vertical inferior
      FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]:=CorNova;  // Atribui nova cor
      inc(auxY);
      until (FFilho.Imagem.Canvas.Pixels[AuxX,AuxY]<>CorAnt);
    end;
  end;
```

5.2.1.5 HISTOGRAMA E REALCE

O histograma é obtido a partir da contagem dos valores vermelho, verde e azul (RGB) para cada *pixel* da figura. O Quadro 8 demonstra este procedimento.

Quadro 8 – Histograma (Contagem de *Pixels*)

```
Var
  TotCorR,TotCorG,TotCorB : array [0..255] of integer;
Begin
{Soma as Ocorrencias de Pontos}
for x:=0 to Fig.Width-1 do
  for y:=0 to Fig.Height-1 do
    begin
      R:=GetRValue(Fig.Canvas.Pixels[x,y]);
      G:=GetGValue(Fig.Canvas.Pixels[x,y]);
      B:=GetBValue(Fig.Canvas.Pixels[x,y]);
      inc(TotCorR[R]);
      inc(TotCorG[G]);
      inc(TotCorB[B]);
    end;
End
```

Para exibir o histograma, um componente `TImage` foi criado com tamanho fixo de 256 *pixels* de largura, onde cada nível de cor (0 a 255) será representado por cada um destes *pixels*. O algoritmo de exibição do canal vermelho é exibido no Quadro 9.

Quadro 9 – Exibição do Histograma

```
if CbVermelho.Checked then
    for i:=0 to 255 do
      begin
        y:=trunc(TotCorR[i]*MaxY/MaxRGB);
        if y>MaxY then Y:=Maxy;
        if y<>0 then
          begin
            IHistor.Picture.Bitmap.Canvas.Pen.Color:=RGB(i,0,0);
            IHistor.Picture.Bitmap.Canvas.MoveTo(i,MaxY);
            IHistor.Picture.Bitmap.Canvas.LineTo(i,MaxY-y);
          end;
      end;
```

Para realçar uma imagem, o protótipo recalcula o canal ou canais selecionados, por percentual ou deslocando. Para o cálculo por percentual o nível do *pixel* para o canal é recalculado com a sua multiplicação por uma taxa percentual, enquanto que o cálculo por deslocamento subtrai (para escurecer) ou soma (para clarear) um valor de deslocamento.

O algoritmo pode tratar todas as cores ou apenas uma faixa de níveis de cor. Para isso ser possível, o processo de calculo só é efetuado para cores que pertencerem ao intervalo informado. O Quadro 10 apresenta o algoritmo de realce do canal vermelho de uma figura.

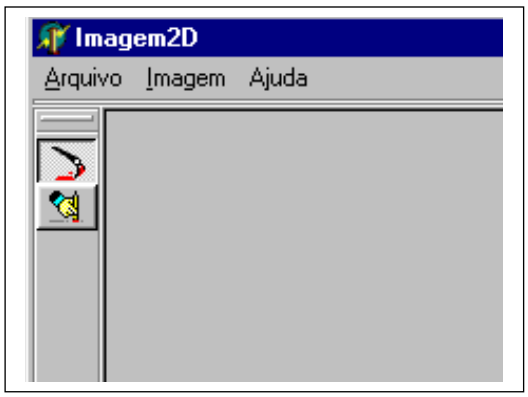
Quadro 10 – Algorítmo de Realce para Canal Vermelho

```
Var
  FaixaI, FaixaF : Integer;   // Faixa inicial e final para realce
  Canal : Byte;               // Canal
  Taxa  : Real;               // Taxa percentual
  Deslocamento : Integer;     // Valor de deslocamento

for y:=0 to fig.Height-1 do
  for x:=0 to fig.Width-1 do

    { Pesquisa toda a figura }

    begin

      Canal:=GetRValue(Fig.canvas.pixels[x,y]);  // Guarda canal vermelho

      if (Canal>=FaixaI) and (Canal<=FaixaF) then  // Se pertence a faixa

        if RbMudarPor.ItemIndex=0 then
          Canal:=Trunc(Canal*Taxa);         // Calculo por percentual

        else

          Canal:=Trunc(Canal+Deslocamento); // Calculo por deslocamento

    end;
```

5.3 FUNCIONAMENTO DO PROTÓTIPO

Nesta seção será descrito o desenvolvimento do protótipo. A tela inicial apresenta as opções Arquivo, Imagem, Janela e Ajuda (Figura 27).

Figura 27 – Menus do Protótipo



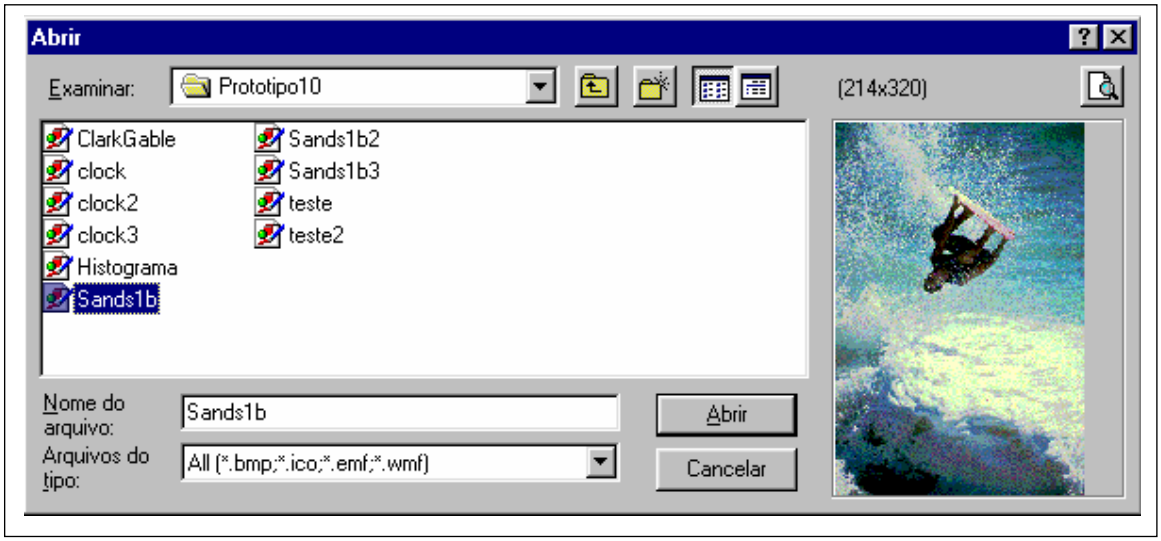
5.3.1 MENU ARQUIVO

O menu Arquivo (Figura 28) apresenta opções relacionados a abertura e salvamento dos arquivos gráficos, bem como a opção de saída do protótipo.

Figura 28 – Opções do Menu Arquivo

As opções Abrir e Salvar Como, quando são acionadas (Figura 29), abrem uma caixa de diálogo onde o usuário poderá selecionar o caminho e a figura a ser aberta ou o novo nome para a figura a ser salva. O protótipo abre e salva figuras nos formatos BMP, ICO, EMF e WMF.

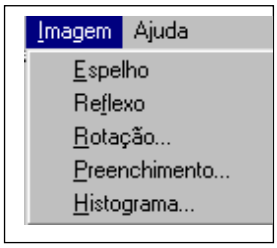
Figura 29 – Caixa de Diálogo do Protótipo



5.3.2 MENU IMAGEM

O menu `Imagem` (Figura 30), possui opções de transformações na imagem. Nesta opção são implementadas as técnicas de Espelho, Reflexo, Rotação, Preenchimento de regiões, Histograma e Realce.

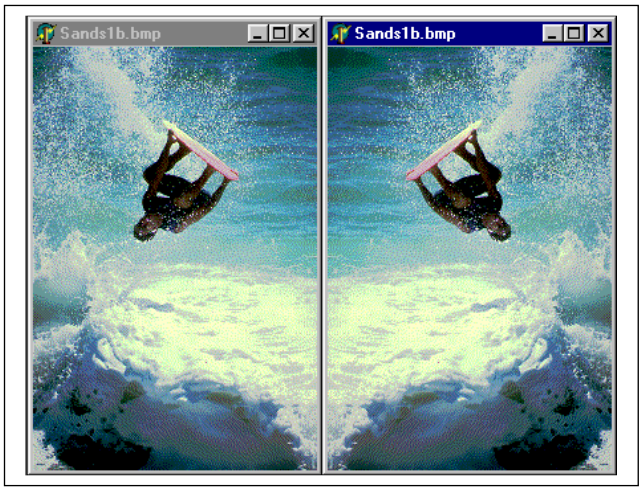
Figura 30 – Opções do Menu Imagem



5.3.2.1 ESPELHO

A técnica de `Espelho` consiste em transformar a imagem em seu espelhamento, ou seja, sendo vista ao contrário (Figura 31).

Figura 31 – Espelho da Imagem



5.3.2.2 REFLEXO

A técnica de `Reflexo` consiste em transformar a imagem em seu reflexo, ou seja, sendo vista de cabeça para baixo (Figura 32).

Figura 32 – Reflexo da Imagem

5.3.2.3 ROTAÇÃO

A técnica de `Rotação`, faz a rotação da figura conforme parâmetros informados pelo usuário do protótipo. A Figura 33 mostra a tela desta técnica.

Figura 33 – Tela de Rotação

O usuário deve informar em quantos graus deseja rotacionar a figura, em qual sentido (horário ou anti-horário), qual cor de fundo assumir e, ainda, qual técnica usar para corrigir os ruídos que podem ocorrer.

As técnicas utilizadas para corrigir ruídos são:

- `técnica 1`: assume para o *pixel* com ruído o mesmo valor do *píxel* imediatamente anterior (x-1);

- `técnica 2`: guarda o valor de cor dos *pixels* que, quando recalculados, iriam sobrepor outros calculados anteriormente. E usa cada um destes valores para pintar a região com ruído;
- `técnica 3`: faz um cálculo da cor intermediária obtida pela média da cor do pixel anterior (x-1) e posterior (x+1) para o ponto que se quer pintar.

Na Figura 34 tem-se um exemplo de rotação de uma figura onde foram informados 45 graus no sentido anti-horário com a seleção da técnica 3 (faz a média de pontos) para correção de ruídos.

Figura 34 – Exemplo de Rotação

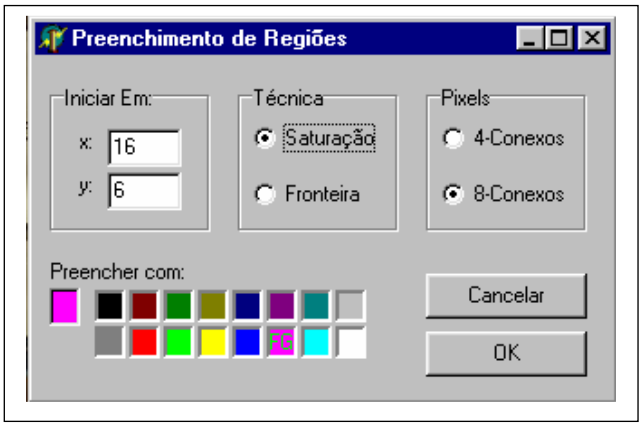
5.3.2.4 PREENCHIMENTO DE REGIÕES

Para atribuir um outro valor de *pixel* para uma determinada região, existe a técnica de preenchimento de regiões. Na tela desta técnica (Figura 35) é disparado o processo de preenchimento para uma determinada região, onde as coordenadas do *pixel* de partida (*germen*) e a técnica a ser utilizada (descritos na seção 4.2) são selecionadas.

No quadro `Iniciar Em`, são informadas as coordenadas `x` e `y` do *pixel* de partida. Já na caixa `Técnica`, será optado pelo tipo de algoritmo e na caixa `Pixels` serão

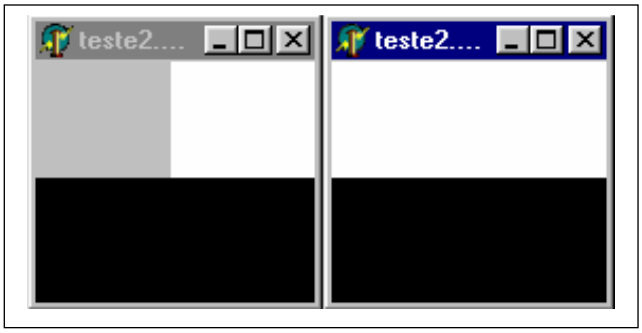
selecionados os *pixels* que serão verificados se `4-conexos` ou os `8-conexos` (descritos na seção 4.2.1).

Figura 35 – Tela de Preenchimento de Região



Todos os pixels pertencentes a região do *pixel* inicial são substituídos Figura 36) pelo que foi selecionado em `Preencher Com`.

Figura 36 – Exemplo de Preenchimento de Região

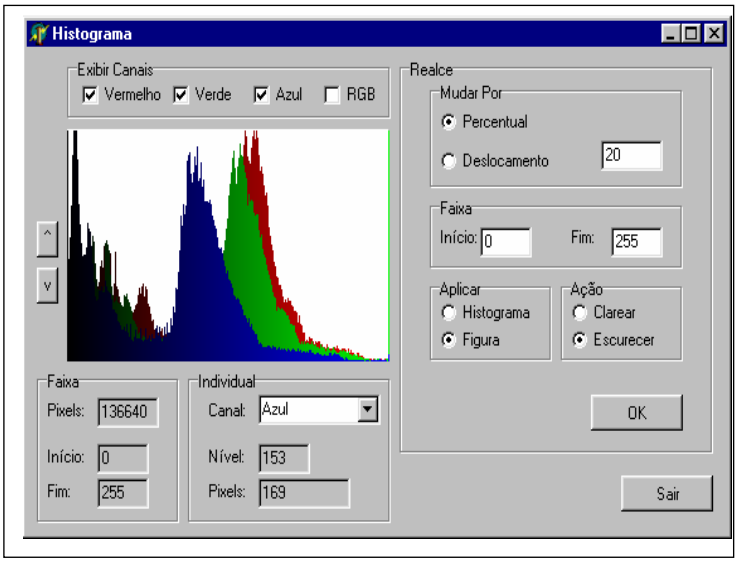


5.3.2.5 HISTOGRAMA

Esta opção permite a visualização dos histogramas da imagem bem como a sua modificação através da técnica de realce da imagem.

O histograma é a representação visual da contagem dos valores RGB para os *pixels* da figura. Os valores de cada canal está representado horizontalmente no histograma (0 a 255) e verticalmente são representadas as contagens que cada um apresentou (Figura 37).

Figura 37 – Tela de Histograma e Realce de Imagem



O usuário seleciona quais canais deseja que sejam exibidos através da opção `Exibir Canais`. São mostradas também informações de quantidade total de *pixels* que a imagem possui, qual é o valor inicial e final de *pixel* e informações individuais de cada canal.

Para realçar uma figura, alguns parâmetros devem ser informados. A opção `Mudar Por` determina o tipo de cálculo que será feito para obter o novo valor para um determinado canal ou mais (conforme os canais selecionados em Exibir Canais). Se a opção Percentual for selecionada o valor informado é tratado como uma valor percentual. Caso a opção `Deslocamento` for selecionada o valor informado é tratado como um valor fixo de deslocamento para escurecer ou clarear a imagem.

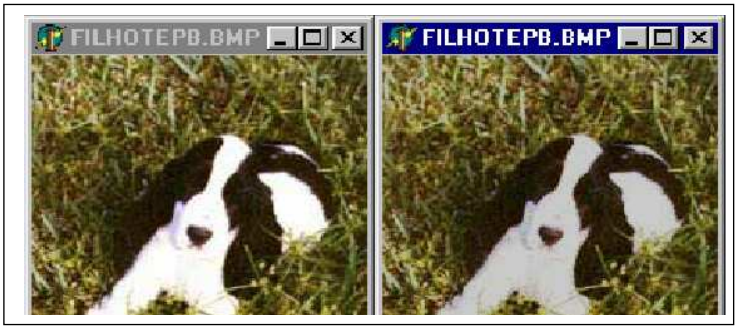
A opção de seleção de faixa, determina quais são os *pixels* para os quais o novo valor será atribuído. O cálculo de realce pode ser efetuado diretamente na figura ou apenas no histograma, conforme opção selecionada em `Aplicar`.

A opção `Ação` determina se o cálculo será feito diminuindo o valor do canal, ou seja escurecendo a figura, ou aumentando o valor do canal clareando a figura. A Figura 38 mostra um exemplo de realce de figura onde os canais vermelho, verde e azul são escurecidos em 60% para toda a faixa de pixels.

Figura 38 – Realce de Figura em Toda a Faixa

A Figura 39 apresenta o realce para uma figura, onde se escureceu os valores RGB de apenas para a faixa mais clara da imagem (de valores 220 até 255) em 30%.

Figura 39 – Realce da Figura na Faixa 220 a 255



Na Figura 40, a imagem original foi escurecida em 10% todos os seus *pixels* para o canal azul, obtendo-se um aspecto envelhecido.

Figura 40 – Realce de Envelhecimento

6 RESULTADOS FINAIS

6.1 CONCLUSÕES

O presente trabalho teve o objetivo de relatar os resultados do estudo sobre técnicas de processamento de imagens para arquivos em formato *raster*. Para que algumas das técnicas estudadas fossem implementadas criou-se um protótipo de um ambiente onde as figuras em formato BMP fossem abertas e permitindo assim, a manipulação dos seus valores de *pixels* através de técnicas implementadas.

As técnicas implementadas no protótipo visam identificar um ou mais algoritmos que podem ser utilizados em imagens 2D para formatos de arquivos *raster*, em processamentos de transformação, edição e realce de imagens.

Entre as técnicas de transformação tem-se a rotação onde pôde-se estudar e implementar formas de tratar os “ruídos” gerados para alguns ângulos.

Em edição de imagens se estudou técnicas para fazer o preenchimento de regiões, onde se implementou e estudou formas diferentes atendendo a necessidade do usuário.

O protótipo utilizou da exibição do histograma da imagem para que o usuário pudesse realçar figuras, aumentando ou diminuindo os valores RGB para níveis e canais diferentes. Conseguiu-se, com isso, criar imagens com características “envelhecidas”, clarear e/ou escurecer partes diferentes de figura, podendo assim melhorar a visualização de figuras que em que uma parte está muito escura e outra parte está muito clara.

As técnica de rotação, preenchimento de regiões e realce da imagem através da modificação de seus *pixels* trouxeram um início a este estudo de processar imagens *raster* 2D, servindo como base para estudos posteriores.

O desenvolvimento deste trabalho alcançou os objetivos finais inclusive com a criação de alguns algoritmos alternativos para a execução de seus processos, tornando este, um diferencial em relação a sistemas existentes no mercado. Contribuindo, assim, para trabalhos futuros que podem ser desenvolvidos na área de transformação de imagens *raster* 2D.

As ferramentas utilizadas foram adequadas principalmente o uso do ambiente `Delphi 5.0` que proporcionou a utilização de alguns conceitos como `MDI` e o uso de componentes gráficos como o `Timage`.

6.2 EXTENSÕES

Como extensão ao protótipo tem-se um abrangente leque de opções para arquivos *raster* 2D.

Podem ser implementadas outras técnicas de realce que efetuam modificações na saturação e nitidez da imagem, técnicas de corte e colagem com seleção através de várias formas como “Varinha Mágica” e “Laço”, técnicas utilizadas para edição da imagem, possibilitando a criação e manipulação de *pixel* a *pixel* como ferramentas Pincel, Borrar e para criação de figuras geométricas.

REFERÊNCIAS BIBLIOGRÁFICAS

[ADO2000] Adobe Systems Incorporated. ***Adobe developers association graphics and publishing SDK***. 2000. Endereço eletrônico: http://partners. Adobe.com/asn/developer/gapsdk/main.html

[BRO1995] BROWN, C. Wayne; SHEPHERD, Barry J. ***Graphics file formats – reference and guide***. Greenwich : Manning Publications Company, 1995.

[CHA1989] CHANG, Shi-kuo. ***Principles of pictoral information systems design***. Englewood : Prentice-Hall International Editions, 1989.

[COR1994] CORRIGAN, John. **Computação gráfica – segredos e soluções**. Trad. de Hugo de Souza Melo, Mariza de Andrade Flores. Rio de Janeiro : Ciência Moderna, 1994.

[FAR1987] FARINA, M. ***Psicodinâmica das cores em comunicação***. Edgard Blücher, 1987.

[FER1986] FERREIRA, Aurélio Buarque de Holanda. Novo dicionário da língua portuguesa. 2ª ed. Rio de Janeiro : Nova Fronteira, 1986.

[GOL1996] GOLDBERG, Walter Neto. **Windows 95**. São Paulo : Érica Ltda, 1996.

[GON1992] GONZALES, Rafael C. WOODS, Richard E. **Processamento de imagens digitais**. Trad. de Roberto Marcondes Cesar Junior. São Paulo : Edgard Blücher Ltda, 1992.

[HAL1988] HALL, Roy. ***Illumination and color in computer genetated imagery***. New York : Springer-Verlay, 1988.

[JAC1994] JACKSON, R***. Computer generated color : a pratical guide to presentation and display***. Jonh Wiley & Sons, 1994.

[LEV1993] LEVINE, John. ***Programing for graphics files in C and C++.*** New York: Wiley, 1993.

[MAG1986] MAGALHÃES, Leo Pini. **Computação gráfica interfaces em sistemas de computação gráfica**. Campinas : Papirus/Unicamp, 1986.

[MAR1987] MARCUS, A. Color : a tool for computer graphics communication. Color in Computer Graphics nº 24 – SIGGRAPH – 1987.

[MCC1999] McClelland, Deke. **Photoshop 5 & 5.5 – bíblia – edição ouro**. São Paulo : Market Books, 1999.

[MEL1990] MELENDEZ FILHO, Rubem. **Prototipação de sistemas de informações: fundamentos, técnicas e metodologias**. Rio de Janeiro : Livros Técnicos e Científicos, 1990.

[MOR1995] MORRISON, Mike. **Mágicas da computação gráfica**. Trad. De Elisa M. Ferreira. São Paulo : Berkeley, 1995.

[PAZ1988] PAZ, Eduardo Peixoto. **Inicialização ao processamento digital de imagens**. Rio de Janeiro: UFRJ/NCE, 1988.

[PAR1993] PARKER, Jim R. *Pratical computer vision using C*. New York : John Wiley, 1993.

[PER1989] PERSIANO, Ronaldo César Marinho; OLIVEIRA, Antônio Alberto Fernandes de. **Introdução a computação gráfica.** Rio de Janeiro : Livros Técnicos e científicos, 1989.

[PLA1991] PLASTOCK, Roy A. KALLEY, Gordon. **Computação gráfica**. Lisboa : McGraw-Hill, 1991.

[ROG1990] ROGERS, David F. ***Mathematical elements for computer graphics***. New York : McGrawHill, 1990 2ª ed.

[RUS1995] RUSS, John C. ***The image processing handbook***. 2ª ed. Boca Raton: CRC Press, 1995.

[SUN1991] SUN Microsystems Inc. An introduction to computer graphics concepts - from pixels to pictures. Carole McClendon, 1991.