Instituto de Matemática e Estatística
Universidade de São Paulo

Visocor
Sistema de Acessibilidade Visual

André Shoji Asato
Rafael de O. Lopes Gonçalves

Orientador: Prof. Dr. Roberto Hirata Jr.

São Paulo, 2009

Resumo

O objetivo deste trabalho é apresentar o desenvolvimento de um software de acessibilidade para pessoas com deficiência visual para cores. Usando técnicas de processamento de imagens em tempo real, um conjunto de filtros provê facilidades no entendimento de conteúdos exibidos em tela para portadores destas deficiências.
Palavras-chave: daltonismo, cores, OpenGL, acessibilidade

Sumário

Parte I

Parte Objetiva

Capítulo 1

Introdução

1.1 Motivação

A motivação deste trabalho tem origem em uma história particular de um dos responsáveis por este trabalho. Ele tentava verificar as condições de fluidez de trânsito no site da Companhia de Engenharia de Trafego de São Paulo (CET) porém não conseguia compreender as informações no mapa: as cores da legenda o confundiam. Isto se devia ao fato de ele ser portador de uma deuteronamalia[1].

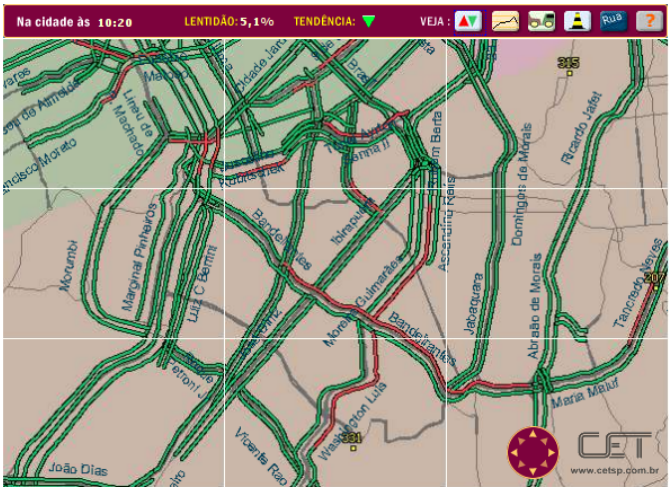
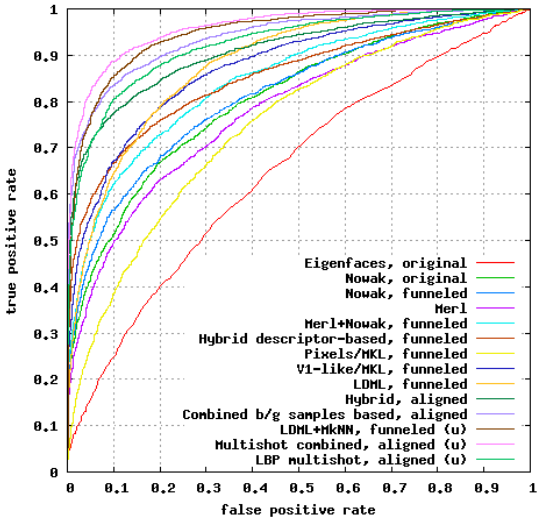


Figura 1.1 – *Mapa de fluidez de trânsito (fonte: www.cetsp.com.br)*

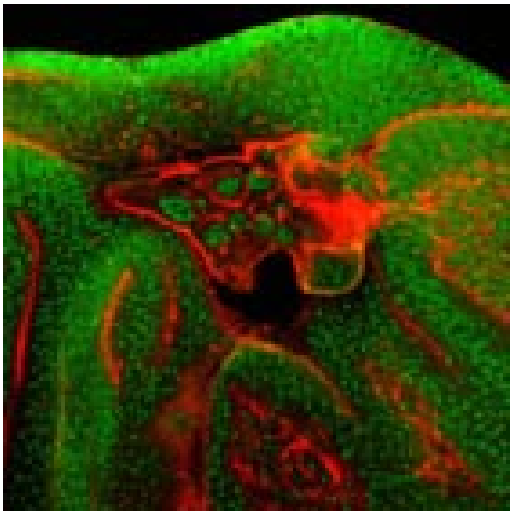
A tarefa que ele não pode realizar consiste na tentativa de perceber corretamente áreas de interesse do mapa com base nas suas legendas. Assim sendo, dado um padrão de cor, apontar na imagem onde este padrão reaparece.

Uma outra possível classe de tarefa que pode causar dificuldade para portadores de deficiência visual cumprirem é compreender imagens quando informações visuais relevantes se baseiam em alguns contrastes de cores não perceptíveis para essas pessoas.

[1] explicamos o significado desta condição no capítulo 2



(a) Informações de alguns gráficos podem depender demais da diferenciação de cores (fonte:[6])



(b) Imagem de célula cujo contraste vermelho/verde é necessário para compreender suas informações relevantes (fonte: [12])

1.2 Objetivos

Atualmente, existe uma preocupação quanto a acessibilidade para diversas deficiências. Apesar dos esforços de projetistas de sites e de interfaces de software em providenciar essa acessibilidade, é muito difícil se adequar a todas as possíveis deficiências.

Por meio de filtros e ferramentas ajustáveis, é possível para o deficiente perceber e diferenciar informações visuais relevantes que normalmente não seriam notadas. Para aplicar este filtro, o usuário necessitaria capturar a imagem e aplicar transformações em ferramenta de edição gráfica. Isto acaba sendo muito dispendioso e repetitivo.

O sistema descrito neste trabalho tem o objetivo de providenciar acessibilidade para deficientes visuais para cores de modo que possam realizar as tarefas descritas na seção anterior com facilidade.

Dada a dificuldade das soluções atuais, este trabalho tem também como objetivo que esta acessibilidade seja possível sob demanda, de maneira pervasiva. Como aplicações diretas podemos citar a visualização de:

- Mapas com legendas baseadas em cores;
- Gráficos com excesso de cores que impossibilitam a diferenciação pelo contraste;
- Sites que desconsiderem deficientes visuais.
- Imagens e fotografias.
- Interfaces de softwares.

1.3 Estrutura deste Trabalho

Neste capítulo mostramos as dificuldades em acessibilidade que este trabalho tem como objetivo de resolver. No capítulo 2 apresentamos os conceitos básicos da visão de cores pelo ser humano, a sua representação digital e a visão deficiente. No capítulo 3 descrevemos e justificamos as tecnologias usadas neste trabalho. Nos capítulos 4 e 5 explanamos as duas técnicas usadas para suprir acessibilidade às tarefas descritas na seção 1.1. E, por fim, apresentamos o resultado final e possíveis estudos futuros no capítulo 6

Capítulo 2

Visão de Cores

Para que possamos estudar os problemas apresentados no capítulo 1, é necessário uma compreensão mínima do que é uma cor, como é percebida pelo ser humano com visão normal e, finalmente, por um ser humano com uma anomalia para percepção de cores.

2.1 Cor

> **cor,** *subst. f.* **1.** Impressão produzida no olho pela luz, segundo a sua própria natureza ou a maneira pela qual se difunde nos objetos. **2.** Aparência dos corpos segundo o modo como refletem ou absorvem a luz.
> ...
> *Dicionário Aurélio da Língua Portuguesa.*

Se a cor é uma impressão, trata-se não de um efeito somente físico mas sim psicofísico e, portanto, relativo ao ser que a sente. Palmer [15] diz que estamos enganados ao dizer que determinado objeto é de certa cor. O que acontece, na verdade, são respostas psicológicas a características físicas dos objetos e das luzes envolvidas.

2.1.1 Caráter Físico da Cor e Luz

Entre 1670 e 1672 Sir Isaac Newton realizou diversas experiências e descobertas no campo da ótica.

Uma dessas experiências consistiu em passar um feixe de luz solar por um prisma. Nesta experiência, Newton verificou a formação de um arco-íris. Com o auxílio de outro prisma, os feixes deste arco-íris recombinaram-se novamente em luz branca. Através deste famoso experimento, Newton constatou que a luz solar é formada por diferentes cores e que a cor não está na própria luz mas sim no efeito por ela criado no sistema visual humano.

> "The Rays to speak properly are not coloured. In them there is nothing else than a certain Power and Disposition to stir up a sensation of this or that Colour. ... So Colours in the object are nothing but a Disposition to reflect this or that sort of Rays more copiously than the rest" *(Isaac Newton)*

As descobertas de Newton foram os primeiros [1] pensamentos sobre as propriedades físicas da luz que produzem a visão de cores. Desde então ocorreram grandes avanços nesta área.

[1] Para uma coleção de referências sobre a teoria das cores de Goethe, a qual é complementar a ótica de Newton ver `http://www.ime.usp.br/~vwsetzer/pals/Goethe.html`

Figura 2.1 – *Capa do disco The Dark Side of the Moon do Pink Floyd com a experiência de Newton*

O fóton, a unidade básica de luz, é um pequeno pacote de matéria vibrante[15]. Este tem comportamento que pode ser caracterizado tanto como o de onda quanto o de partícula. Para o estudo de cores, o comportamento do fóton como onda é o mais importante.

Como a velocidade da luz é constante, é possível caracterizar um feixe de luz somente pelo seu comprimento de onda e a quantidade de "pacotes" (fótons) por unidade de tempo. Os possíveis comprimentos de onda dos fótons variam de alguns nanômetros a vários quilômetros. Contudo, dentre todas as possíveis variações do comprimento de onda da luz, somente um pequeno intervalo produz experiência de luz visível. O espectro de luz visível pelos seres humanos está entre 400 e 700 nm. São exemplos de luz não visíveis os raios x, micro-ondas e sinais de TV e rádio.[2]

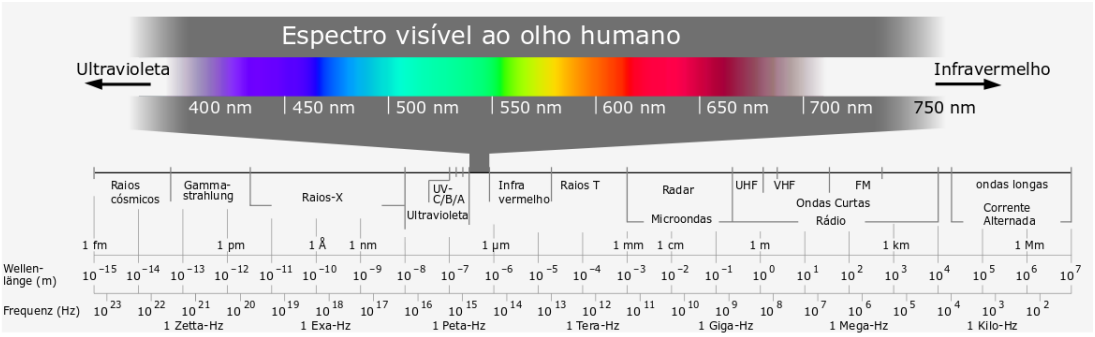


Figura 2.2 – *Pequena parte das ondas eletromagnéticas que é visível*

2.2 Sistema Visual Humano

Um sistema visual fornece a um organismo a capacidade de enxergar. Tal capacidade é obtida ao captar e interpretar as informações luminosas visíveis de modo a construir uma representação própria do mundo ao redor.

[2]figura 2.2 traduzida do original de Horst Frank, Jailbird and Phrood: spektrum-elektromagnetischer-wellen, sob licença GFDL

No casos dos seres humanos, os órgãos (e uma explicação bastante simplificada de suas funções) que constituem o sistema visual são:

Olho

Orgão que detecta a luz visível. Das estruturas que formar o olho, destacamos a retina, um tecido fotossensível localizado no fundo da superfície interna do olho.

Nervo Óptico

Transmite a informação visual da retina para o cérebro.

Quiasma Óptico

Parte do cérebro onde os nervos ópticos se cruzam. É onde metade das informações visuais captadas pelos olhos trocam o lado de destino no cérebro, isto é, as informações captadas pelo olho direito destinam-se parcialmente ao lado esquerdo do cérebro e vice-versa. A outra metade das informações não sofre este cruzamento.

Trato Óptico

Continuação do nervo óptico que segue até o cérebro após o quiasma óptico.

Núcleo Geniculado Lateral

Localizado no tálamo do cérebro, é o processador primário das informações visuais.

Radiação Óptica

Coleção de axônios dos neurônios associativos que transmitem as informações visuais do tálamo para o córtex visual.

Córtex Visual

Parte do cérebro responsável por processar as informações visuais e transmitir os resultados para serem interpretados.

Córtex de Associação Visual

Interpreta os estímulos visuais obtidos do processamento das informações e nos dá a percepção de como vemos o mundo.

Para este projeto, nos focamos nos estágios iniciais da percepção visual humana. A etapa em questão encontra-se na captura dos raios de luz visíveis que atingem os olhos. Em particular, no comportamento de um componente localizado no interior do olho responsável pelos primeiros procedimentos de percepção visual de cores.

2.2.1 Bastonetes e Cones

Bastonetes e cones são células fotorreceptoras localizadas na retina.

Os bastonetes estão mais concentrados na parte periférica da retina e são mais sensíveis à luz que os cones. Devido à essa alta sensibilidade tornam-se os principais fornecedores de informações

visuais sobre a intensidade de luz presente, principalmente em casos de pouca luz. Sendo assim, os bastonetes também são responsáveis pela visão noturna.

Os cones, por sua vez, apresentam maiores concentrações de células na parte central da retina e, se comparadas com os bastonetes, são menos sensíveis à luz mas permitem a percepção de cor.

Elas também percebem melhor os detalhes e as mudanças rápidas das imagens pois suas respostas aos estímulos são mais rápidas que a dos bastonetes.

Um elemento importante presente nas células cone são os pigmentos fotossensíveis (fotopigmentos). Cada fotopigmento reage a comprimentos específicos de onda de luz e estimulam as células cone de acordo com esse comprimento de onda. Cada célula cone possui uma quantidade diferente e diferentes tipos de fotopigmentos. No caso dos seres humanos, pela quantidade de fotopigmentos, agrupa-se as células cones em três tipos, o que caracteriza uma visão tricromática de cores, denominadas:

- L (de *long*): respondem à luz de comprimentos de onda mais longos, onde o ápice é a área da cor amarela (por volta de 564 a 580 nm).
- M (de *medium*): respondem à luz de comprimentos de onda médios, onde o ápice é a área da cor verde (por volta de 534 a 545 nm).
- S (de *short*): respondem à luz de comprimentos de onda curtos, onde o ápice é a área da cor violeta (por volta de 420 a 440 nm).

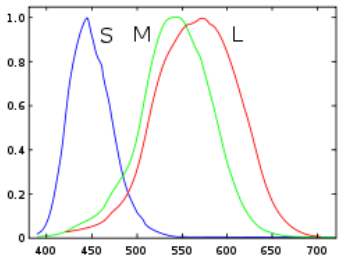


Figura 2.3 *– Intensidade de estímulo percebida por tipo de células cone dependendo do comprimento de onda da luz.*

Assim, a cor amarela/esverdeada é percebida ao estimular levemente mais os cones L do que os M; a cor vermelha é percebida ao estimular muito mais os cones L do que os M; e as cores azul/violeta são percebidas ao estimular mais os cones S do que os outros dois.

2.3 Colorimetria e Padronização

Colorimetria[19] é o ramo da ciência que estuda o problema de como definir numericamente o estímulo de cor de maneira que:

- sob mesmas condições, estímulos sob a mesma especificação parecem iguais
- estímulos que parecem iguais tem a mesma especificação

- os números da especificação são funções dos parâmetros físicos

A partir daí supõe-se a generalização tricromática: o estímulo de diversas cores pode ser correspondido com a mistura aditiva de três estímulos primários.

Qualquer conjunto de estímulos onde um não pode ser simulado pela mistura de outros dois pode servir como três estímulos primários. Ou seja, cada estímulo, em termos vetoriais, é linearmente independente entre si, formando uma base tridimensional.

No modelo de cores RGB, os estímulos primários são os de cor vermelho, verde e azul. A escolha dessas cores se deve ao comportamento físico do olho humano caracterizado pela visão tricromática de cores.

Este modelo é o mais comumente utilizado em dispositivos de visualização (monitores, televisores, etc) e é a representação do espaço de cores utilizada pelas tecnologias envolvidas neste projeto.

2.4 Sistema Visual Humano deficiente

As definições da padronização tricromática assumem, para os resultados obtidos na mistura de cores, o visualizador como uma pessoa com visão normal para cores. Mas, na prática, isso não ocorre pois uma parcela da população possui algum tipo de deficiência na visualização de cores.

Aproximadamente 8% da população masculina e em torno de 1% da feminina possui esta característica[15]. Essas pessoas não têm capacidade de distinguir todas as cores da gama visível em relação à visão normal. O problema está na ausência de um ou mais tipos de células cone ou na anomalia de algum tipo de fotopigmento. Esta ausência está relacionada, majoritariamente, a uma herança genética ligada o alelo X, o que explica a maior quantidade de homens portadores da deficiência.

É comum o portador não ter consciência de sua condição. Para tal, existem testes específicos como, por exemplo, o teste de Ishihara que consiste em uma série de cartões impressos com círculos de diversas cores, formando na imagem números os quais só são vistos completamente por uma pessoa com visão normal. No cartão de exemplo da figura 2.4 o número 74 deve ser visível para pessoas com visão normal. Pessoas com alguma deficiência podem enxergar o número 21 ou até mesmo nenhum número.

Indivíduos com visão normal são chamadas de tricromatas, uma vez que possuem os três tipos de células cone, seus fotopigmentos não apresentam anomalias e, assim, podem perceber corretamente todas as cores a partir dos três estímulos fundamentais.

Aqueles que apresentam ausência de somente um tipo de cone são chamados de dicromatas. Há também tricromatas possuidores de anomalias, chamados de tricromatas anômalos. Nesse caso, a percepção de um dos três estímulos é anômala porém ainda presente. Eles podem criar correspondência de qualquer cor usando os três estímulos mas respondem a estes de maneira diferente da normal.

Dadas as condições acima apresentadas, os indivíduos nessas condições podem apresentar os

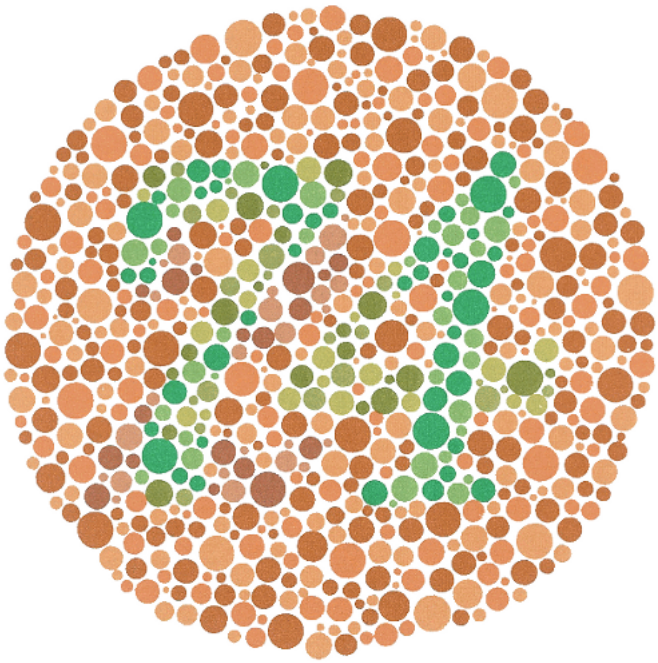


Figura 2.4 – *Exemplo de cartão de teste de Ishihara.*

seguintes sintomas:

Impossibilidade de distinguir contrastes vermelho/verde

Sintoma comum a indivíduos portadores de protanopia, protanomalia, deuteranopia ou deuteranomalia. Nessas condições, não é possível distinguir pequenas diferenças entre as cores da seção vermelho/verde do espectro de cores.

O prefixo *proto*, no caso, se refere ao fato de o problema ser na percepção do primeiro estímulo fundamental, o da luz de cor vermelha. Esse radical ainda define um grupo – *protan* – que engloba a protanopia e a protanomalia. Analogamente, *deutero* refere-se para a luz de cor verde e *deutan* é o grupo que contém a deuteranopia e a deuteranomalia.

A protanopia e a deuteranopia estão relacionadas com a não funcionalidade ou total ausência de células cone do tipo L e M, respectivamente. Ademais, protanópicos e deuteranópicos confundem luz de comprimento de onda próximos de 492 e 498 nm, respectivamente, como luz branca. Os protanópicos confundem também a cor vermelho escuro com a cor preta.

A protanomalia e a deuteranomalia estão ligadas com alguma anomalia nos fotopigmentos sensíveis à luz de comprimentos de onda mais longos e médios, respectivamente. A deuteranomalia é a anomalia é a mais comum nos casos de deficiência na percepção de cores.

Impossibilidade de distinguir contrastes azul/amarelo

Sintoma comum a indivíduos portadores de tritanopia ou tritanomalia. Nessas condições, não é possível distinguir as pequenas diferenças entre as cores da seção azul/amarelo do espectro de cores.

Analogamente às condições anteriores, o prefixo *trito* refere-se ao terceiro estímulo fundamental, o da luz de cor azul, e tritanopia e tritanomalia – agrupadas em *tritan* – estão relacionadas à ausência de células cone do tipo S e alguma anomalia nos fotopigmentos sensíveis à luz de comprimentos de onda mais curtos, respectivamente.

Diferentemente das demais deficiências, ambas não são hereditariamente relacionadas ao sexo.

Por ser a deficiência com maior número de ocorrências e, também, por ser aquela que um dos dos integrantes deste projeto possui (deuteranomalia), concentramos os estudos e trabalhos sobre a incapacidade de distinguir contrastes vermelho/verde.

2.5 Observação

Para fins práticos, a partir de agora o termo cor é citado neste trabalho referindo-se ao vetor com três valores das intensidades das cores espectro fundamental vermelho, verde e azul, nesta ordem.

Capítulo 3

Tecnologias Usadas

3.1 X Window System

X Window System, mantido pela *X.Org Foundation* e comumente chamado de X ou X11, é um gerenciador de interfaces gráficas do usuário (GUIs) altamente configurável, compatível com diversas plataformas e gratuito. Os sistemas operacionais Unix em geral e Linux em várias de suas distribuições existentes adotam o X como o sistema de janelas padrão.

O X utiliza um modelo cliente-servidor peculiar: cada máquina local contém um servidor X que se comunica local ou remotamente com diversos clientes X.

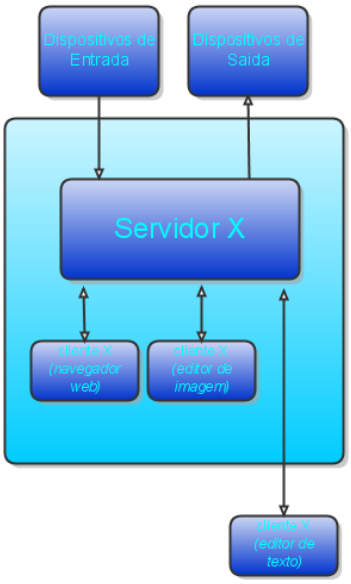


Figura 3.1 – *Arquitetura cliente-servidor do X*

3.1.1 Gerenciador de Janelas

Gerenciador de janelas é um sistema que controla o posicionamento das janelas dentro de um sistema de janela. Eles podem ser implementados de modo a exibir diretamente na tela o

conteúdo de cada janela ou ainda de modo a redirecionar o conteúdo de cada janela para buffers individuais.

Para a exibição dos dados na tela, o X necessita de um gerenciador de janelas.

3.1.2 *Composite manager*

Composite manager é um dos componentes da GUI que possui acesso aos buffers mencionados anteriormente. O *composite manager* processa, manipula e combina o conteúdo desses buffers entre si. O resultado final desses buffers, chamado de *pixmap*, fica disponível para serem, posteriormente, desenhados por meio de primitivas OpenGL[1] pelo *composite manager*. Desse modo, cada a saída de cada aplicação comporta-se como objetos 2D ou 3D. Isto possibilita a criação de efeitos como, por exemplo, os da figura 3.4.

3.2 OpenGL

OpenGL, *Open Graphics Library*, é um conjunto de especificações e definições que padroniza uma *API*[2] multiplataforma, multilinguagem para a criação de aplicações gráficas 2D e 3D.

3.2.1 Linguagem de *shader*

O processamento tradicional gráfico feito pelas implementações da OpenGL inclui um uma sequência rígida de funções que são executadas. Esta sequência, chamada de *pipeline*, pode ter parâmetros alterados, ou partes desativadas e reativadas mas mantem-se inalterada. Uma versão (muito) simplificada desta sequência tradicionalmente realizada ([14] e [18]) é descrita a seguir.

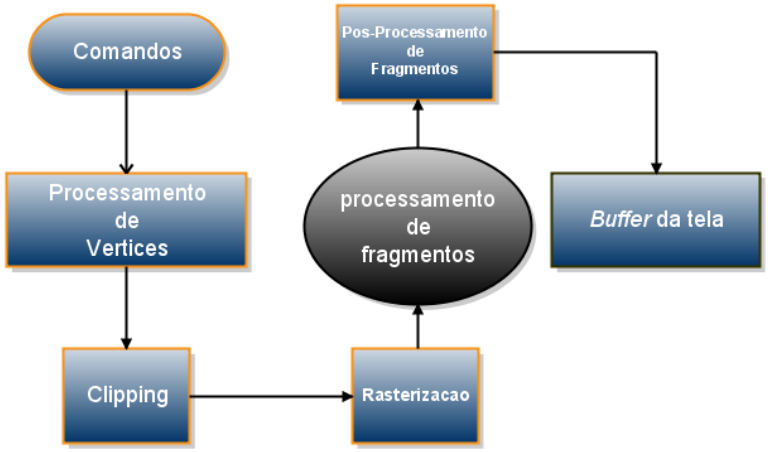


Figura 3.2 – Pipeline *simplificado da OpenGL*

Processamento de Vértices

Realiza processamento dos vértices (criados com `glVertex*`, por exemplo) aplicando a

[1] isto ocorre no caso do Compiz

[2] *API: application programming interface* – um conjunto de instruções e rotinas encapsuladas de modo a facilitar a implementação de algum outro software

matriz de transformação *Model-View* para realizar rotações, translações etc., monta as primitivas (triângulos, quádricas, NURBS, etc.) e faz cálculos de iluminação e sombras.

Clipping

Calcula quais vértices são exibidos e quais estão fora da área de exibição cortando polígonos onde for necessário e/ou criando vértices novos.

Processamento de Fragmento

Os processos anteriores criam dados chamados fragmentos que contém os valores interpolados dos vetores normais, cor primária e secundária e coordenada da textura a ser usada. Nesta etapa é calculada a cor final usada na posição de cada fragmento.

Pós-processamento dos *pixels*

Após a avaliação das cores e posição de cada *pixel*, esta etapa realiza o processamento de anti-serrilhamento, o teste de transparência e o de profundidade.

***Buffer* da Tela**

Depois de todo o processamento, os dados são colocados em um espaço especial da memória e finalmente estes podem ser exibidos pelo sistema gráfico.

Em versões mais novas de OpenGL (a partir da versão 1.5) tanto a etapa de processamento de vértices quanto a de fragmentos pode ser substituída por um programa chamado de programa de *shader* de Vértice e de Fragmento respectivamente.

Assim, é possível usar o poder de processamento vetorial e de ponto flutuante das *GPU* para aplicações mais específicas do que as de uso geral criada para o fluxo de dados de OpenGL.

Esses *shader* s têm uma linguagem de programação própria e são compilados em tempo de execução da aplicação OpenGL. Além disso, são implementados como extensões a OpenGL e portanto podem ou não serem implementados pelos dispositivos gráficos.

Inicialmente, o X previa apenas a execução de aplicativos em 2D. No entanto, sua arquitetura possui um mecanismo de extensões que permite a adição de complementos ao seu protocolo. Com isso, pode-se acrescentar ao X suporte a aplicações com efeitos gráficos 3D.

3.3 GLX: OpenGL no X

GLX (*Open**GL** Extension to the **X** Window System*) visa o uso do OpenGL dentro do X. Essa extensão do X consiste em uma *API* fornecedora de funções OpenGL ao X e em extensões adicionais ao cliente e servidor. Com essas extensões o cliente envia, junto com os comandos X padrões, comandos de renderização 3D ao servidor e o servidor recebe esses comandos, reinterpreta os comandos OpenGL e os envia para a biblioteca OpenGL. Esse mecanismo é conhecido por renderização indireta [3].

[3] do inglês *indirect rendering*

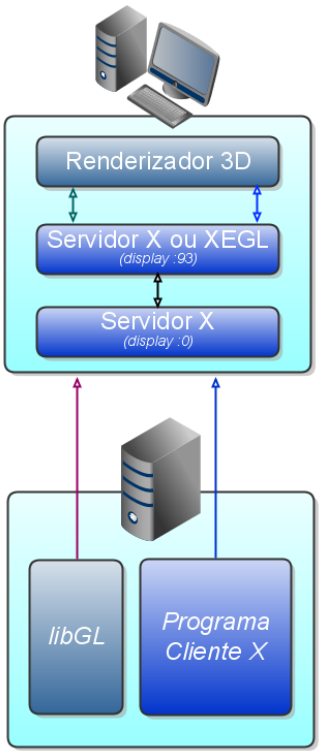


Figura 3.3 – *Arquitetura XGL*

3.3.1 Aceleração 3D por dispositivo gráfico

Com vista na popularização das placas gráficas com aceleração 3D e para aproveitar ao máximo suas capacidades, era ainda necessário replanejar a arquitetura do X que já se encontrava defasada e não possuía suporte para aceleração via hardware.

Xgl

É um servidor X reescrito cuja arquitetura forma uma camada sobre o OpenGL com suporte para aceleração gráfica via hardware para todas as aplicações X e OpenGL. O Xgl é apenas uma interface, ou seja, necessita de uma implementação para funcionar.

O Xglx é uma implementação desenvolvida para o Xgl. Ela cria, pela interface GLX, uma janela com suporte OpenGL em um servidor X. O Xgl, então, utiliza essa janela. Na prática, é um servidor X funcionando sobre outro.

AIGLX (Accelerated Indirect GLX)

É um servidor X levemente alterado, com inclusão de extensões próprias que permite renderização indireta e acelerada. Utiliza-se também da interface GLX.

Tanto o Xgl quanto o AIGLX dependem da extensão OpenGL texture_from_pixmap (`GLX_EXT_texture_from_pixmap`) cujo suporte suporte é fornecido pelos

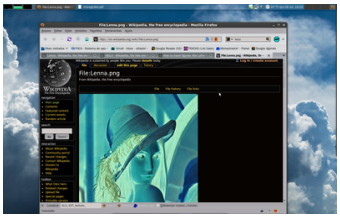
drivers de vídeo. Essa extensão transforma pixmaps em texturas as quais podem ser processadas como em qualquer aplicação gráfica que usa texturas OpengGL.

Atualmente, o Xgl encontra-se defasado e o AIGLX está oficialmente incorporado ao X.

3.3.2 Compiz

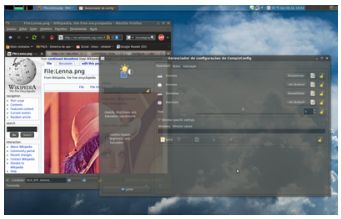
Compiz[17] é um *compositioning window manager* (*composite manager* + gerenciador de janelas) escrito na linguagem de programação C e com código-fonte aberto que utiliza tanto o Xgl quanto o AIGLX. O Compiz é extensível por meio de *plugins*. Desse modo, funcionalidades inicialmente não fornecidas pelos *plugins* oficiais são simplesmente incorporáveis.

Desenvolvido junto ao Xgl, o Compiz sofreu uma ramificação, conhecida por Beryl. Mais tarde, o Compiz dividiu-se em Compiz-Core, mas com o nome Compiz mantido, e em Compiz-Extra, sendo este fundido ao Beryl, resultando no Compiz-Fusion. O Compiz sofreu mais ramificações: Nomad, enfocado principalmente no acesso remoto, e o Compiz++, uma versão do Compiz diferenciada portada para linguagem de programação C++. Por fim, tem-se a fusão do Compiz, Compiz-Fusion, Nomad e Compiz++ novamente mantendo o nome Compiz.



(a) Negativo

(b) Cubo



(c) Transparência

Figura 3.4 *– Efeitos proporcionados pelo Compiz*

Ferramentas utilitárias do Compiz

Uma ferramenta importante no desenvolvimento de *plugins* para o Compiz é o BCOP, abreviação de *Beryl/Compiz Options Parser*. O BCOP gera automaticamente o código-fonte em C de configuração de um *plugin* à partir de um arquivo XML. Uma vez compilado junto ao código-fonte do *plugin* e instalado, as opções de configuração tornam-se disponíveis no CCSM.

O CCSM, abreviação de *CompizConfig Settings Manager*, é a ferramenta de configuração do Compiz usada para ajustar o comportamento e o funcionamento dos seus *plugins*. Ele apresenta uma interface gráfica ao usuário para configuração a partir do código gerado pelo BCOP.

3.4 As escolhas de tecnologia para o projeto Visocor

Visando processar os gráficos em tempo real e de maneira pervasiva ao usuário, implementamos o software **Visocor** como um plugin para o Compiz. Desta maneira pode-se acioná-lo para filtrar o conteúdo de qualquer janela no sistema de janelas do X11.

O processamento do filtro e do destacador de cores descritos a seguir são realizados em contexto OpenGL. Sendo assim acelerados pelo dispositivo de processamento gráfico deixando o processador de uso geral livre.

O projeto de software deste trabalho usa em sua implementação um *shader* de Fragmento que desenvolvemos especificamente para realizar as tarefas de processamento gráfico do **Visocor**.

Para este projeto, foi usada uma linguagem básica de montagem de *shader* de fragmento, muito semelhante a uma linguagem de montagem de processadores de uso geral. Porém, as instruções são relacionadas ao processamento vetorial e de ponto flutuante comumente usados no contexto destes *shaders* [2]. Usando uma linguagem antiga e mais simples, esperamos permitir que uma gama maior de dispositivos de processamento gráficos, mesmo os mais antigos, possam realizar o processamento dos gráficos do software.

Capítulo 4

Filtro

Dado o problema de identificar contrastes não visíveis para pessoas protanomalas ou deuteranomalas, criamos uma heurística de transformação da das imagens exibidas ao usuário. Esta heurística de agora em diante será chamada simplesmente de filtro.

O filtro, cujo cerne é implementado usando-se a linguagem de *shader* s de OpenGL, baseia-se na ideia de que contrates entre tons de verde e vermelho são difíceis ou impossíveis de serem percebidos por protanomalas ou deuteranomalas. Já os contrastes verde/azul e vermelho/azul são mais facilmente notados.

Sendo assim, a intenção da heurística do filtro é aumentar a intensidade da cor azul em lugares que contenham altas intensidades de verde tornando os contrastes vermelho/verde contrastes vermelho/azul. A mesma técnica é aplicada ao vermelho criando contrastes verde/azul.

4.1 A heurística do filtro

O filtro, através de uma matriz de transformação $\mathcal{M}$ aplicada sobre a cor de cada *pixel* da imagem, troca os contrastes imperceptíveis por perceptíveis.

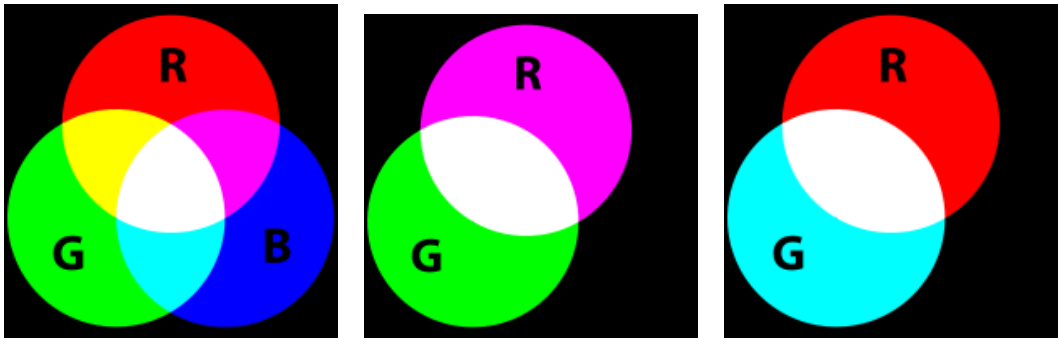
Algoritmo 4.1.1: Heurística do Filtro Matricial

Entrada: Conjunto de *pixels* da tela

1 **para cada** pixel p **faça**
2 $\quad R \longleftarrow p_{vermelho}$;
3 $\quad G \longleftarrow p_{verde}$;
4 $\quad B \longleftarrow p_{azul}$;
5 $\quad [R, G, B]^T \longleftarrow [R, G, B] \cdot \mathcal{M}$;
6 $\quad p_{vermelho} \longleftarrow R$;
7 $\quad p_{verde} \longleftarrow G$;
8 $\quad p_{azul} \longleftarrow B$;
9 **fim**

Onde $\mathcal{M}$ é a seguinte matriz 3×3

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & \alpha \\ 0 & 1 & \beta \\ 0 & 0 & \gamma \end{pmatrix} \tag{4.1}$$



(a) Paleta de cores vermelho-verde-azul sem alterações

(b) Paleta de cores passada pelo filtro com $\alpha = 1$ e $\beta = 0$

(c) Paleta de cores passada pelo filtro com $\alpha = 0$ e $\beta = 1$

Figura 4.1 *– Paleta de cores vermelho-verde-azul sofrendo alterações pelo filtro*

e α, β e γ são tais que:

$$\alpha = \begin{cases} 0 & \text{se } \textit{intensidade} \geq 0; \\ -\textit{intensidade} & \text{se } \textit{intensidade} < 0. \end{cases} \tag{4.2a}$$

$$\beta = \begin{cases} 0 & \text{se } \textit{intensidade} \leq 0; \\ \textit{intensidade} & \text{se } \textit{intensidade} > 0. \end{cases} \tag{4.2b}$$

$$\gamma = 1 - (\alpha + \beta) \tag{4.2c}$$

O valor de *intensidade*, por sua vez, é tal que $\textit{intensidade} \in [-1, 1]$ e é controlado pelo usuário através de um comando simples de mouse ou teclado.

Desta forma, a matriz $\mathcal{M}$ é controlada interativamente pelo usuário através de uma única variável, *intensidade*, definindo qual contraste é modificado (verde ou vermelho) e o quanto é modificado.

Por exemplo, quando $\textit{intensidade} < 0$ temos que $\alpha > 0$, $\beta = 0$ e, portanto, a transformação realizada pela matriz $\mathcal{M}$ aumenta a intensidade da cor azul em *pixels* que já possuem a cor vermelha.

4.1.1 O porquê de se usar uma matriz de transformação

Uma maneira de se re-escrever o filtro descrito pelo algoritmo 4.1.1 não usando a matriz de transformação é descrito no algoritmo 4.1.2.

Algoritmo 4.1.2: Heurística do Filtro modificada

Entrada: Conjunto de *pixels* da tela, α, β

1 **para cada** pixel p **faça**

2 $\quad p_{azul} \longleftarrow (p_{azul} \cdot \gamma) + (p_{vermelho} \cdot \alpha) + (p_{verde} \cdot \beta)$;

3 **fim**

A abordagem matricial tem vantagem sobre esta última, dado o conjunto de instruções disponibilizado pela linguagem de *shader* de OpenGL [2].

Capítulo 5

Destaque de cores

O filtro apresentado no capítulo anterior ajuda a solucionar o problema onde é necessário visualizar contrastes entre cores que são difíceis para pessoas com protanomalia e deuteranomalia perceberem. Porém, nem sempre o difícil é visualizar contrastes. Algumas vezes a dificuldade do usuário é encontrar cores semelhantes em regiões não vizinhas. Um exemplo, citado no capítulo 1, é o caso onde temos um gráfico com diversas legendas baseadas em cores e deseja-se encontrar a curva correspondente.

A curva não faz fronteira com a legenda e nem necessariamente com outras curvas. Deseja-se de alguma maneira fazer com que a curva que corresponde a determinada legenda seja destacada das outras. Em suma, queremos que, dada uma cor x, todas as ocorrências dela na imagem sejam destacadas. O pensamento inverso também é possível: ocultamos todos os lugares onde não ocorre a cor x. Logo, temos o seguinte algoritmo:

Algoritmo 5.0.3: Destaque da cor I

Entrada: Conjunto de *pixels* da tela, e uma cor x

```
1 para cada pixel p faça
2 |  se x ocorre em p então
3 |  |  Destacar p;
4 |  senão
5 |  |  Ocultar p;
6 |  fim
7 fim
```

O algoritmo 5.0.3 deixa três lacunas a serem preenchidas. A primeira é o que é x ocorrer em um *pixel*. A segunda e a terceira são como se destaca e/ou se oculta um *pixel*.

5.1 O que é uma cor ocorrer em um *pixel*

A solução trivial é afirmar que se a cor de p é igual a cor x, então x ocorre em em p:

$$\big(x \quad \text{ocorre em} \quad p\big) \Leftrightarrow \Big((p_{vermelho} = x_{vermelho}) \wedge (p_{azul} = x_{azul}) \wedge (p_{verde} = x_{verde}) \Big) \tag{5.1}$$

No entanto, esta solução é ingênua pois, como uma imagem pode ter pequenas variações

na cor devido a edições da imagem, ruídos, melhoramentos prévios de anti-serrilhamento, entre outros, a condição $cor(p) = x$ pode ser muito radical, não selecionando pequenas variações da de x.

Uma generalização desta solução que nos dá uma família de possíveis abordagens é calcular a distância entre a cor de p e a cor x. Se esta distância for pequena o bastante, dizemos que x ocorrem em p.

$$x \text{ ocorre em } p \quad \Leftrightarrow \quad distancia(x, cor(p)) < \delta \tag{5.2}$$

Dois tipos de cálculo de distância foram considerados para este trabalho: a distância de Chebyshev e distância euclidiana. Sejam x e y duas cores:

- **Distância Euclidiana:** É a norma da diferença de x e y.

$$distancia_{euclidiana}(x, y) = \sqrt{\left(\sum_i [x_i - y_i]^2\right)} \quad , \quad i \in vermelho, verde, azul. \tag{5.3}$$

- **Distância de Chebyshev:** [1] Consiste em somente considerar a maior diferença entre as coordenadas vermelho, verde e azul das cores.

$$distancia_{Chebyshev}(x, y) = \max\left(|x_i - y_i|\right) \quad , i \in vermelho, verde, azul. \tag{5.4}$$

5.1.1 Escolha do limite máximo da distância

Tendo em vista que os software tem o objetivo de prover um ambiente de acessibilidade personalizável, o valor de δ é controlado pelo usuário. E mais, é desejável que, conforme a necessidade do usuário, a seleção seja mais precisa para diferenças de intensidade de verde do que vermelho ou azul e vice-versa. Portanto, δ deve ser um vetor de três dimensões especificando a precisão para cada eixo: $\delta = (\delta_{vermelho}, \delta_{verde}, \delta_{azul})$.

Uma maneira de se usar estes três valores é pensando na forma geométrica do sólido criado pelo conjunto de cores selecionadas para um dado valor $\mu \in \mathbb{R}$ usando-se a distância euclidiana e a distância Chebyshev.

5.1.2 A solução: uma elipsoide

Observando a equação 5.3 notamos que o conjunto de cores que serão selecionadas por um dado μ forma uma esfera no espaço de cores com raio μ. De maneira análoga, usando-se a distância de Chebyshev, o conjunto de possíveis cores a serem selecionadas para dado μ forma um cubo cujas arestas têm comprimento μ. Ou seja, para as cores x e y:

$$Selecionados^x_{euclidiana} = \{y \mid distancia_{euclidiana}(x, y) \leq \mu\} \tag{5.5a}$$

$$Selecionados^x_{Chebyshev} = \{y \mid distancia_{Chebyshev}(x, y) \leq \mu\} \tag{5.5b}$$

[1] a distância de Chebyshev também é conhecida como distância tabuleiro pois representa quantos passos no mínimo o rei de um jogo de xadrez precisa realizar para alcançar uma casa

Para levar em consideração os valores do vetor δ na distância euclidiana e Chebyshev, as formas geométricas poderiam formar elipsoides e paralelepípedos em vez de esferas e cubos respectivamente.

Usando a equação cartesiana do elipsoide e do paralelepípedo formulamos o seguinte:

$$Selecionados^{x}_{euclidiana} = \left\{ y \mid \sum_{i} \frac{(xi - yi)^2}{\delta_i^2} \leq 1 \right\} \tag{5.6a}$$

$$Selecionados^{x}_{Chebyshev} = \{ y \mid |x_i - y_i| \leq \delta_i \} \tag{5.6b}$$

onde $i \in \{vermelho, verde, azul\}$.

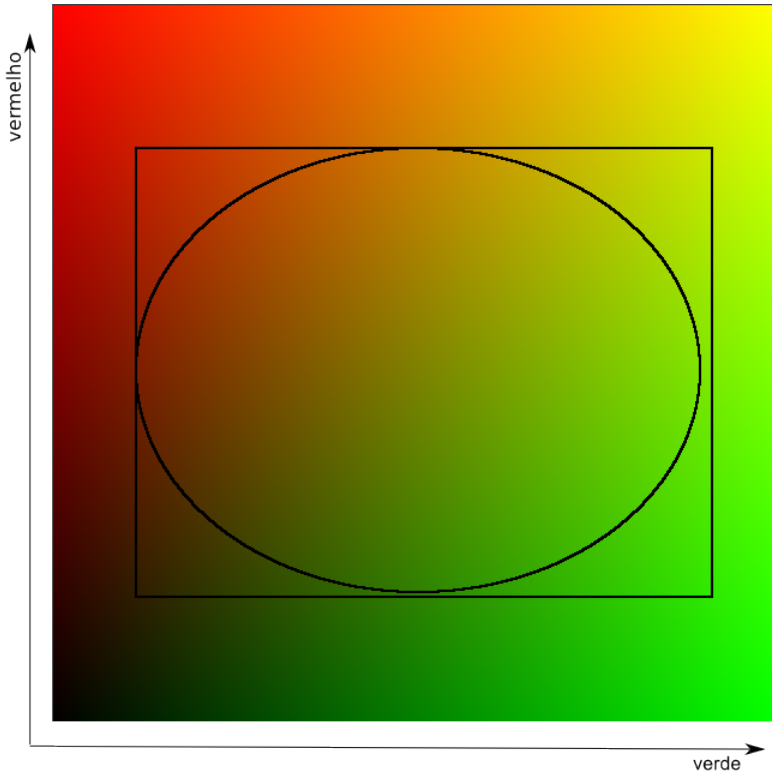
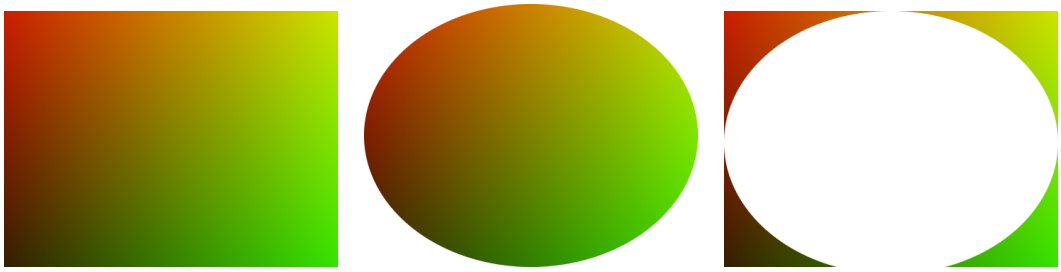


Figura 5.1 – *Plano azul* $= 0$ *com as seleções de Chebyshev e euclidiana*

A solução que usa o elipsoide fornece uma seleção mais apurada que a solução do paralelepípedo. Isto se deve ao fato da existência da área referente aos cantos do paralelepípedo. Os valores nessas áreas extremas ficam mais longes do centro do sólido em mais do que somente uma das dimensões. Isso que não acontece nos extremos do elipsoide.

O conjunto de figuras 5.2 mostra um possível corte dos sólidos sobrepostos. A figura 5.2(c) mostra as cores selecionadas pelo paralelepípedo mas não selecionados pelo elipsoide.

Revisitando o algoritmo 5.0.3 com a formula da elipsoide, temos:



(a) Seleção usando distância de Chebyshev

(b) Seleção Usando distância euclidiana

(c) Diferença entre Selecionados pela distancia de Chebyshev e euclidiana

Figura 5.2 *– Cortes no plano azul* $= 0$ *mostrando seleções usando os mesmos parâmetros*

Algoritmo 5.1.1: Destaque de cor II com elipsoide

Entrada: Conjunto de *pixels* da tela, uma cor x, e um vetor δ de três elementos: vermelho, verde e azul

```
1 para cada pixel p faça
2     se ∑_i (xi−cor(p)_i)^2 / δ_i^2 ≤ 1 então
3         Destaca p;
4     senão
5         Oculta p;
6     fim
7 fim
```

5.2 Destacando e Ocultando uma cor

Uma vez separadas as cores a serem selecionadas, é necessário, de alguma forma, ocultar as cores não selecionadas e destacar as selecionadas (fora e dentro do elipsoide, respectivamente). É desejável ainda que as cores alteradas não alterem demais a imagem de maneira que, após a aplicação, ainda seja possível distinguir as formas que estão na imagem.

Então, criamos arbitrariamente a seguinte heurística: as cores selecionadas não são alteradas e as não selecionadas sofrem o seguinte processo:

Transformada em tom de cinza: o valor de cada elemento da cor é alterado para a média simples dos valores anteriores. Ou seja:

$$\text{cinza\_x}_i \longleftarrow \frac{\sum_i x_i}{3}, i \in \{vermelho, verde, azul\} \tag{5.7}$$

Obtenção do negativo: à partir da cor em tom de cinza, obtemos o valor complementar, ou

seja, se somarmos ambos os valores, temos a cor branca absoluta[2]:

$$\text{negativo\_x} \longleftarrow \text{branco} - \text{cinza\_x} \tag{5.8}$$

Sendo assim, apresentamos a versão revisada e final do algoritmo 5.0.3.

Algoritmo 5.2.1: Destaque de cor III, final

Entrada: Conjunto de *pixels* da tela, uma cor x, e um vetor δ de três elementos: vermelho, verde e azul

```
1  para cada pixel p faça
2      se Σ_i ((xi − cor(p)_i)^2 / δ_i^2) ≤ 1 então
3          Mantém p inalterado;
4      senão
5          para cada i ∈ {vermelho, verde, azul} faça
6              cinza_p_i ⟵ (Σ_i cor(p)_i) / 3 ;
7              negativo_x ⟵ branco − cinza_x ;
8              cor(p)_i ⟵ negativo_p_i ;
9          fim
10     fim
11 fim
```

[2]em OpenGL a cor branca é representada pelo valor 1.0 nos elementos vermelho, verde e azul da cor

Capítulo 6

Conclusão

6.1 Resultados

O **Visocor** é um software cuja finalidade é facilitar ou permitir a percepção do contraste entre tons verde e vermelho normalmente desapercebido completamente em pessoas protanomalas ou deuteranomalas. Concebemos o software como um *plugin* open-source para o Compiz escrito, por esse motivo, na linguagem de programação C. O **Visocor** utiliza as *APIs* fornecidas pelo Compiz e pelo OpenGL que permitem seu funcionamento em diversas plataformas e sistemas operacionais.

Por ser um *plugin* para o Compiz, o **Visocor** realiza o processamento gráfico em tempo real e executa de modo pervasivo ao usuário.

O funcionamento do **Visocor** se dá em parceria com o do Compiz. Este permite ao nosso software a manipulação, sobretudo, da aparência das janelas dos aplicativos em execução. A parte importante a ser utilizada pelo **Visocor** são as cores que estas janelas apresentam.

(a) Tela de configurações do seletor de cores

(b) Tela de configurações com as teclas de atalho do software

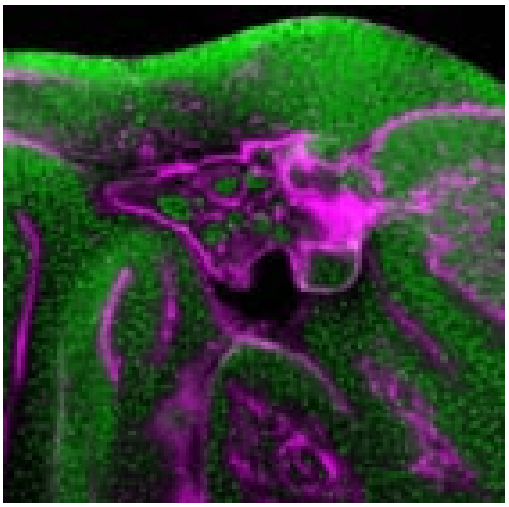
Figura 6.1 *– Telas de configurações e ajustes de parâmetros*

O **Visocor** aplica um filtro para cores heuristicamente desenvolvido por nós. O filtro utiliza a linguagem de
shader do OpenGL e é parametrizável. Assim, seu ajuste é feito conforme a necessidade do usuário. Este ajuste, por padrão, é feito com um combinação de teclas mais a roda do mouse.

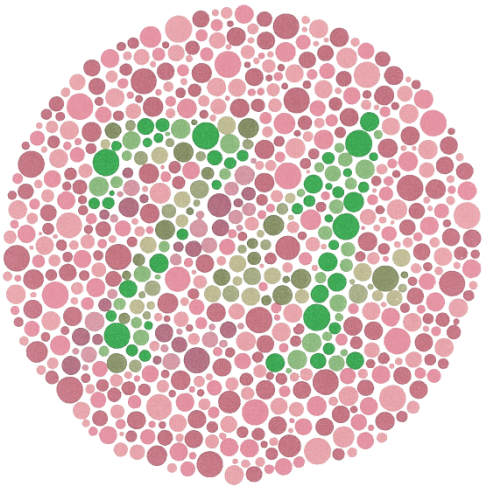
As cores dos *pixels* são modificados proporcionalmente pelo filtro conforme o seu valor de vermelho e verde, aumentando o azul. Dessa forma, o contraste vermelho/verde passa a ser um

contraste azul/verde ou azul/vermelho onde não há dificuldade ou impossibilidade de distinguí-los.

Com este efeito, é possível resolver a tarefa de identificar contrastes como o da figura 8.1(b).



(a) Imagem da figura 8.1(b) sob efeito do filtro



(b) Paleta de teste de Ishihara sob efeito do filtro

Figura 6.2 – *Resultados da aplicação do filtro de cores*

Com o contraste agora perceptível, o usuário pode observar as fronteiras de cores mostrados nas janelas das aplicações que antes não seriam possíveis de serem notadas.

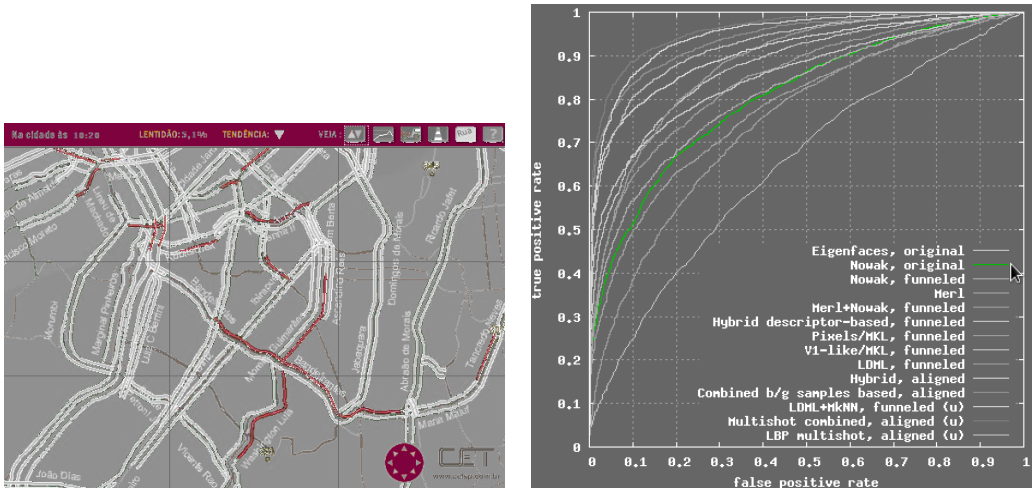
O filtro permite ainda o destaque de uma cor em particular da janela ao pressionar uma combinação de teclas e um clique do mouse. Como nem toda imagem apresentada está livre de ruídos, permite-se incluir pequenas variações de cores junto com a cor indicada (tons semelhantes) pelo cursor do mouse por meio de um limiar de cores ajustável. A cor destacada e seus tons semelhantes mantém-se originais enquanto as demais são mostradas como suas respectivas negativas.

6.2 Trabalhos Futuros

O trabalho atual concentrou-se na criação de um filtro voltado para a protanomalia e deuteronomalia. Para outras anomalias, é necessário adaptar o filtro. Faltou também para o nosso trabalho verificá-lo testando-o com mais pessoas e, assim, qualificar estatisticamente a eficácia do filtro.

O espaço de cores RGB é definido a partir da premissa de uma visão tricromata. Usar um espaço de cor definido sobre a premissa de uma visão dicromata para perceber onde não existem contrastes para uma pessoa com visão anômala poderia ser uma melhoria ao software.

Com o Compiz pode-se aplicar os efeitos dos *plugins* baseados em diversos contextos. Por exemplo: título da janela de aplicação.



(a) Imagem de mapa de fluidez da CET com cor de legenda selecionada

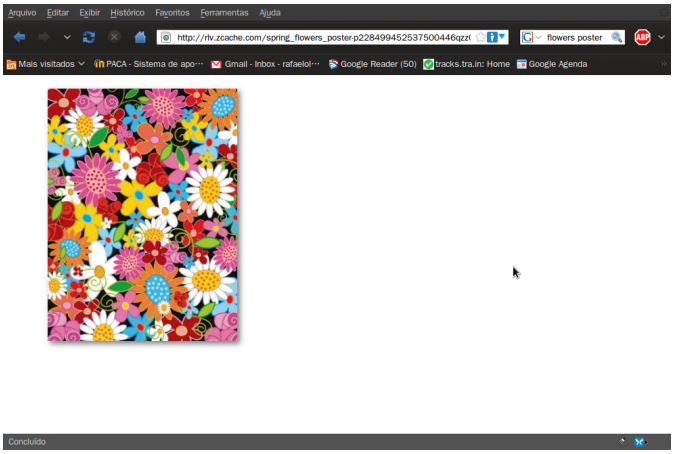
(b) Imagem da figura 8.1(a) com cor de um dos itens da legenda selecionada

Figura 6.3 – *Resultados da aplicação do seletor de cores*

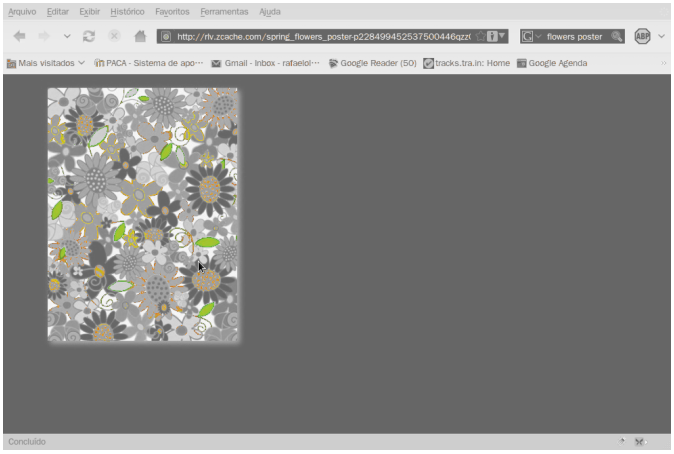
Pode-se, assim, pré-determinar como o **Visocor** deve agir com base na satisfação de algum critério aplicado a algum contexto exemplificado.

Caso de uso: Sabe-se que certos formatos de imagens são mais propensos ou não a possuir ruídos. Logo, é possível pré-determinar um limiar adequado com base nessa informação e aplicá-lo diretamente para a janela da aplicação exibindo a imagem em questão.

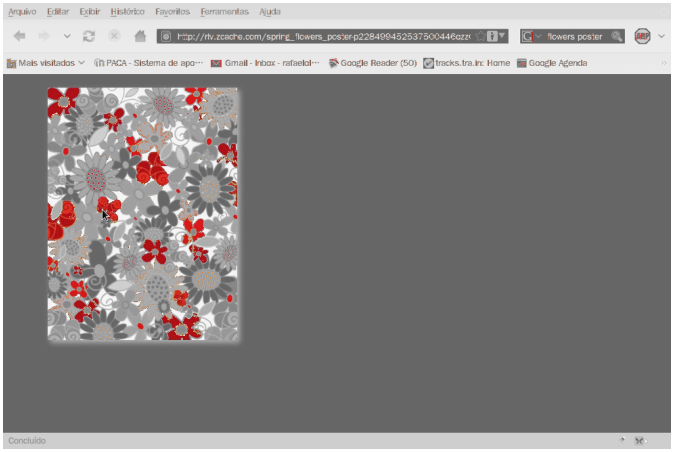
Outra possibilidade seria adaptar as técnicas descritas neste trabalho para uma implementação em algum dispositivo móvel com câmera integrada de maneira que a a acessibilidade provida não se restrinja à tela do computador e possa ajudar o deficiente em diversos momentos.



(a) Imagem sem aplicação do filtro



(b) Seleção de cores sobre as folhas



(c) Seleção de cores sobre as petalas vermelhas

Figura 6.4 – *Seleção de diferentes cores da mesma imagem*

Referências Bibliográficas

[1] Compiz. http://www.compiz-fusion.org/.

[2] OpenGL Architecture Review Board. Arb-fragment-program extension specification, November 2006. http://www.opengl.org/registry/specs/ARB/fragment_program.txt.

[3] OpenGL Architecture Review Board. Glx_ext_texture_from_pixmap extension specification, Fevereiro 2009. http://www.opengl.org/registry/specs/EXT/texture_from_pixmap.txt.

[4] Aurélio Buarque de Holanda Ferreira. *Dicionário Aurélio da Língua Portuguesa*. Editora Nova Fronteira, 1993.

[5] Claudia Feitosa-Santana et al. Espaço de cores, 2006.

[6] Tamara Berg Gary B. Huang, Manu Ramesh and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[7] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[8] Matthias Hopf. compiz: The next generation desktop, 2007. http://www.vis.uni-stuttgart.de/~hopf/pub/LinuxTag2007_compiz_NextGenerationDesktop_Paper.pdf.

[9] Keith Packard James Gettys. The (re)architecture of the x window system, 2004. http://keithp.com/~keithp/talks/xarch_ols2004/xarch-ols2004-html/.

[10] Edward Macnaghten. Accelerated x flame wars!... maybe not, 2006. http://www.freesoftwaremagazine.com/articles/accelerated_x.

[11] Laurence T. Maloney. *Color Vision: From Genes to Perception*, chapter Physics-based approaches to modeling surface color perception. Cambridge University Press, 1999.

[12] Kei Ito Masataka Okabe. Color universal design (cud): How to make figures and presentations that are friendly to colorblind people, 2008. http://jfly.iam.u-tokyo.ac.jp/color/.

[13] Saul Merin. *Inherited eye diseases: diagnosis and clinical management*. Informa Health Care, 1991.

[14] Opengl, D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, August 2005.

[15] S. E. Palmer. *Vision Science-Photons to Phenomenology*. MIT Press, Cambridge, MA, 1999.

[16] The Linux Information Project. The x window system: A brief introduction, 2006. `http://www.linfo.org/x.html`.

[17] The Free Enciclopedia Wikipedia. Wikipedia, the free enciclopedia, 2009. `http://en.wikipedia.org/`.

[18] Richard S. Wright, Benjamin Lipchak, and Nicholas Haemel. *OpenGL(R) SuperBible: Comprehensive Tutorial and Reference (4th Edition) (OpenGL)*. Addison-Wesley Professional, June 2007.

[19] Günther Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae (Wiley Series in Pure and Applied Optics)*. Wiley-Interscience, 2 edition, August 2000.

Parte II

Parte Subjetiva

Capítulo 7

André Shoji Asato

Este trabalho teve início em 2008. Sua motivação foi a dificuldade do Rafael em entender o mapa de congestionamento do trânsito no site da CET. Mesmo podendo entender o mapa, achei bastante interessante a ideia de criar uma ferramenta de acessibilidade e decidi trabalhar com ele sobre este assunto.

O primeiro passo foi fazer uma "pesquisa de mercado". Encontramos softwares disponíveis capazes de realizar aquilo que tínhamos em mente. Contudo, tais soluções eram comerciais (pagos), possuíam código-fonte fechado, eram bastante intrusivos ao usuário (é necessário manter a janela da aplicação aberta e movê-la sobre a área de interesse) ou com uso muito limitado (restritos apenas ao navegador da internet). Também verificamos o processo de carregar as imagens em editores e aplicar diversos filtros sobre elas e o quão trabalhoso isto era. Com isso em mente, acreditamos que criar esta ferramenta de acessibilidade, gratuita e com código-fonte aberto seria algo extremamente significativo.

Apresentamos essa proposta ao professor Hirata que imediatamente achou o tema interessante e aceitou ser nosso orientador.

Com os problemas das soluções existentes, concordamos também que nosso software deveria ser o mais leve possível e pervasivo ao usuário. Para tal, a ideia inicial era que o **Visocor** deveria ser um software em nível de *driver* de vídeo. Isto tornaria o projeto bastante complexo, pois dependeria profundamente dos modelos das placas de vídeo e limitaria drasticamente o número de prováveis usuários.

Dentre os modos de implementar o **Visocor** sem abdicar das condições impostas por nós mesmos, optamos por escrever nosso software como um *plugin* do Compiz. Para os sistemais operacionais Linux, o Compiz é um marco para os chamados *desktops* 3D, sendo seu carro-chefe o famoso *plugin* que torna a área de trabalho em um cubo rotacionável.

Desde quando instalei o Compiz em minha máquina, achei bastante interessante e divertido o efeito visual que ele traz consigo. No entanto, após a euforia inicial, percebi o quão escasso eram as aplicações realmente *úteis*. Assim, o **Visocor** provém mais uma realização: tornar o Compiz um pouco mais útil.

Este projeto foi bastante instrutivo para mim pois pude compreender melhor como a percepção de cores pelas pessoas varia bastante.

7.1 Desafios e frustrações

Felizmente, os códigos-fonte dos *plugins* oficiais do Compiz são abertos. Infelizmente, esses códigos-fonte são a única documentação oficial de como escrever *plugins*. Assim, foi necessário que estudássemos a arquitetura do Compiz e a sua comunicação com e entre os *plugins* com base apenas nestes códigos-fonte, aproveitando somente as funcionalidades de interesse de outros *plugins* e criando, claro, as ausentes. Em contrapartida, o Compiz possui diversas ferramentas auxiliares para o desenvolvedor.

Como sou um tricromata normal, não pude efetivamente testar e nem vivenciar o pleno funcionamento do filtro. Isso, sem dúvidas, é bastante frustrante... Obviamente, me refiro ao

fato de ser eu capaz de perceber informações visuais sem o auxílio do **Visocor**. Posso ver os efeitos que filtro em funcionamento causam mas não posso compartilhar diretamente a felicidade do Rafael em verificar que realmente havia algo "oculto" na imagem.

Minha maior decepção do projeto foi não termos sido capazes de encerrá-lo e de entregá-lo em 2008. No meu caso, não pude conciliar as atividades acadêmicas com o trabalho e o desenvolvimento do **Visocor**. Decidimos então adiar o projeto para o ano seguinte.

A principal meta de 2009 foi não repetir os erros de 2008. Reorganizamos nossas prioridades e metas e seguimos o cronograma da proposta à risca. Os eventos mais favoráveis deste ano também nos ajudaram e assim foi possível finalizar este projeto. Olhando o lado positivo desse atraso de um ano, pudemos desenvolver e entregar um software muito mais maduro e robusto se comparado ao que tínhamos a um ano atrás.

7.2 Disciplinas Relevantes

MAC0110 - Introdução à Computação
Disciplina base do BCC, onde tudo começou. É bastante interessante observar o contraste dos EPs feitos nessa época com o que sei sabe fazer agora.

MAC0122 - Princípios de Desenvolvimento de Algoritmo
Passado o período de adaptação do curso, é nessa disciplina que percebi que não devemos nos apegar muito na linguagem de programação, mas sim na lógica, independentemente da linguagem.

MAC0323 - Estruturas de Dados
Disciplina extremamente importante e muito significativa para as futuras disciplinas. Nela, vi as várias maneiras de como manter e onde armazenar os dados que precisamos operar.

MAC0211 - Laboratório de Programação I
Tive meus primeiros contatos com ferramentas de desenvolvimento utilizadas neste trabalho: Subversion, LaTeXe Make. É também a primeira disciplina em que é necessário o trabalho em equipe para realizar um projeto relativamente grande e o envio de relatórios periódicos sobre o seu desenvolvimento.

MAC0300 - Métodos Numéricos da Álgebra Linear e
MAT0139 - Álgebra Linear para Computação
Em ambas, aprendi os fundamentos de vetores, espaço vetorial e transformações lineares, tópicos que foram utilizados para a representação de cores e na criação do filtro.

MAC0417 - Visão e Processamento de Imagens
Matéria em que estudei conceitos e técnicas que foram aplicadas na criação do filtro.

7.3 Futuro do Visocor

Conforme descrito no capítulo 6.2 (Trabalhos Futuros), nosso principal foco foi no contraste vermelho/verde. O estudo poderia estender também para as demais deficiências visuais quanto a cores existentes não concentradas nesse trabalho.

Um maior amadurecimento no desenvolvimento do **Visocor** poderia nos levar a propor a sua inclusão ao repositório oficial de *plugins* do Compiz. Desse modo, o **Visocor** não evoluiria somente por nós, mas em um ambiente colaborativo. Contudo, seria necessário adequar melhor nosso código às normas exigidas pelos mantenedores do Compiz.

Apesar de ter código-fonte aberto, o Compiz encontra-se ainda bastante restrito ao Linux. Com a continuidade dos estudos, poderíamos nos concentrar mais sobre o funcionamento dos sistemas operacionais e de seus gerenciadores de janelas e considerar também a implementação do **Visocor** como uma aplicação *standalone*.

7.4 Agradecimentos

Agradeço ao professor Hirata pela orientação durante esses dois anos de projeto, aos meus amigos de curso por todos os galhos quebrados, em especial ao Rafael, por me aguentar durante esses 6 anos que estivemos cursando o BCC, aos meus professores ao longo do curso e, finalmente, aos meus familiares, que estão sempre por perto dando aquela força!

Capítulo 8

Rafael de Oliveira Lopes Gonçalves

Durante o ano de 2009 tive a oportunidade de redescobrir um interesse antes perdido pelos estudos do curso de ciência da computação . O fato de eu ter me dedicado exclusivamente ao curso, parando de trabalhar no estágio talvez esteja bastante relacionado ao retorno do interesse.

Este trabalho é uma das coisas mais gratificantes que fiz durante curso de graduação. Primeiramente pois aprendi muito sobre conduzir um trabalho acadêmico com este e sobre trabalhar em grupo. Depois o tema do trabalho, que foi uma ótima escolha. Graças a esta escolha, lidamos com as duas áreas de computação que mais me interesso: visão computacional e computação gráfica. Pudemos desenvolver algo novo, que usa os conteúdos das disciplinas cursadas e ainda pode ser útil para outras pessoas.

O tema deste trabalho foi fomentado por diversas dificuldades que costumo ter ao usar um computador e ver mapas da CET , ao assistir alguma aula em que o professor insiste em usar aquele giz vermelho "invisível" e ao comprar roupas.

Figura 8.1 *– Ao comprar um tênis verde sempre leve um amigo com visão normal*

No começo de 2008 o que era frustração tornou-se estímulo para desenvolver este trabalho. Logo na primeira conversa com o professor Hirata, ele aceitou a proposta de desenvolver este trabalho. Então começamos a pesquisar alternativas de implementação, soluções existentes e o problema de visão deficiente em si.

Apesar de nunca termos parado com a pesquisa do projeto e desenvolvimento, infelizmente não conseguimos concluí-lo no ano de 2008. Falta de tempo, de maturidade, excesso de ego por não querer terminar um trabalho "de qualquer jeito", os motivos são vários.

Felizmente no ano de 2009 começamos cedo. Em Janeiro já estávamos desenvolvendo o software e coletando os dados e referências para escrever a monografia. No primeiro semestre uma parte considerável deste software já estava tão madura que utilizei como projeto para a disciplina de Visão e Processamento de Imagens. O que aconteceu de Julho em diante foi um tranquilo e constante desenvolvimento do software e escritura desta monografia.

Durante este tempo me tornei testador e desenvolvedor ao mesmo tempo, sendo eu e meu irmão [1] as principais "cobaias" do software. Penso que teríamos muito mais dificuldade para testar o software sem esta conveniência.

8.1 Disciplinas Relevantes

Todo o ciclo básico de "MATs" "MAEs": Cálculo I e II, Estatística I e II, Álgebra Linear (esta foi especialmente útil para este trabalho), foram extremamente importantes para construir uma maneira lógica de se pensar. São ferramentais e fundamentais para podermos entender todo o resto do curso. Mas quatro disciplinas tiveram especial importância para a realização deste trabalho.

Álgebra Linear Todas as transformações e operações que criamos usam o fato de uma cor ser um ponto num espaço vetorial de três dimensões. Com os conhecimentos desta disciplina e outras como MAC300 e Programação Linear tivemos maturidade para poder desenvolver os algoritmos deste trabalho.

Visão Computacional e Processamento de Imagens Nos deu fundamentos para poder criar os filtros que alteram as imagens. Parte do **Visocor** foi usado como projeto para esta disciplina.

Computação Gráfica Toda a arquitetura do OpenGL e das técnicas de visualização e renderização nos deram capacidade de fazer diversas escolhas durante o projeto.

Sistemas Operacionais Para o desenvolvimento do software deste trabalho foi necessário ter um entendimento claro sobre o que acontece numa arquitetura Linux com o X. Além disso foi nesta disciplina que perdi um certo medo que alguém pode ter ao se deparar com grandes códigos em linguagens de baixo nível como os projetos de software aberto como Linux e o Compiz. Ou seja, perder o medo de sujar as mãos, de pensar "Sim o sistema todo vai travar. não existem meios de rodar um *debug* e nem saída fácil em terminal, o código é enorme, mal documentado e complexo mas eu sei o que estou fazendo... ou pelo menos não tenho medo de fazê-lo"

8.2 Continuação do Trabalho

O próximo passo deste trabalho é ser integrado à distribuição oficial do compiz e por seguinte ser distribuído em conjunto. Desta maneira, terá um alcance maior e possivelmente maior impacto positivo.

Outras melhorias possíveis sobre este trabalho são os descritos na seção 6.2.

8.3 Considerações finais

Agradeço à indústria cafeicultora por me ajudar a ficar acordado até mais tarde fazendo este trabalho. Obrigado governo do Estado de São Paulo pela minha educação desde a primeira série do ginásio até o último ano da faculdade.

Nestes seis anos na graduação adquiri uma certa maturidade que com certeza não tinha no começo do curso. Aprendi que na maioria das vezes vale a pena escutar certas coisas de pessoas certas . Penso que todo aluno que consegue se graduar na Universidade de São Paulo, em especial no IME, acaba adquirindo uma habilidade de ser auto-didata, como uma condição *sine qua non*. Isso acaba sendo extremamente positivo, apesar de desgastante.

[1] daltonismo é genético :-)

8.3.1 Agradecimentos

Agradeço a alguns professores cuja dedicação aos alunos, cada um à sua maneira, faz toda diferença. Ao professor Carlos Eduardo Ferreira que através de comentários meio sarcásticos sempre deixa escapar um conselho. Ao professor João Eduardo Ferreira por se preocupar e saber que são os alunos com problemas que precisam de orientação. Ao professor Setzer pois tenho certeza que não foi coincidência eu largar meu emprego para me dedicar integralmente ao curso no semestre seguinte ao que cursei a disciplina que ele ministra. E finalmente professor Roberto Hirata, por dar um voto de confiança e ter sido ótimo professor durante todos estes anos.

Tenho que agradecer aos amigos do curso: Bento, Dani, Flavia, Gustavo, Lucas, Melissa, Otranto, Pedro, Samuel, ao amigo e parceiro de trabalho Shoji e ao meu grupo de teatro por me ajudar como puderam. Obrigado por me suportarem pessoas.

E mais importantemente à minha família. Meu irmão Daniel que serviu de cobaia para o trabalho . E finalmente aos meus pais os quais me são como uma bússola indicando a direção certa e um porto seguro nas horas de dificuldades. Obrigado por serem tão compreensivos, vocês são ótimos.