Universidade Federal de Alagoas

Programa de Pós-Graduação em Matemática

DISSERTAÇÃO DE MESTRADO

Métodos de Renderização Não-fotorrealística

Daniel Nicolau Brandão

Rio São Francisco

MATEMÁTICA
A ciência
do infinito

Maceió
Fevereiro de 2008

Daniel Nicolau Brandão

Métodos de Renderização Não-fotorrealística

Dissertação apresentada ao Programa de Pós-graduação em Matemática do Instituto de Matemática, da Universidade Federal de Alagoas - UFAL , como um dos pré-requisitos para obtenção do grau de Mestre em Matemática.

Maceió - Alagoas
2008

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária Responsável: Helena Cristina Pimentel do Vale

B817m Brandão, Daniel Nicolau.
Métodos de renderização não-fotorrealística / Daniel Nicolau Brandão. – Maceió, 2008.
44 f. : il.

Orientador: Adelailson Peixoto.
Dissertação (mestrado em Matemática) – Universidade Federal de Alagoas. Instituto de Matemática. Maceió, 2008.

Bibliografia: f. 42-44.

1. Matemática aplicada. 2. Computação gráfica. 3. Renderização de modelo 3D. I. Título.

CDU: 51:004.92

Daniel Nicolau Brandão

Métodos de Renderização Não-fotorealistica

Dissertação de Mestrado na área de Computação Gráfica submetida em 29 de fevereiro de 2008 à Banca Examinadora, designada pelo colegiado do Programa de Pós-Graduação em Matemática da Universidade Federal de Alagoas, como parte dos requisitos à obtenção do grau de mestre em Matemática.

Prof. Dr. Adelailson Peixoto
UFAL
(Orientador)

Prof. Dr.Vinicius Melo
UFAL

Prof. Dr. Dimas Martínez Morera
IMPA

Agradecimentos

Agradeço ao mestre dos mestres: Jesus. Porque Dele, por Ele e para Ele são todas as coisas.

Agradeço também a minha esposa, aos meus pais, minha irmã e meus sogros pela paciência, cuidado, amor, incentivo e dedicação. Amo vocês.

Ao meu orientador Adelailson Peixoto por toda a paciência, dedicação e por todo conhecimento passado durante esses meses de trabalho intenso.

A André Pizzaia, Marcius Petrúcio, Leonardo Carvalho e José Borges pelo companheirismo durante esses anos de pelaja.

A todos os alunos do mestrado em Matemática da UFAL que de alguma forma contribuiram com a minha formação e esse trabalho.

Ao Douglas, Leandro, Alan, Allyson, Renata e Pedro pela ajuda na implementação dos resultados.

Aos professores Hilário Alencar, Marcos Petrúcio, Krerley Oliveira e Amauri Barros pelas valiosas aulas em que contribuiram muito para minha formação.

Aos professores Vinícius Melo e Dimas Martinez pelas valiosas correções da minha dissertação.

E a CAPES pelo apoio financeiro.

A minha esposa amada Glínia, meus pais Francisco e Cleide,
minha irmã Fabiana e meus Sogros Guido e Zuleide.

RESUMO

Nesta dissertação são discutidos os principais conceitos envolvidos nas técnicas de renderização não-fotorrelística e propõe um esquema geral para implementação de tais técnicas. É discutido também um estilo de renderização não-fotorrealística de desenhos de linha para modelos 3D apresentado por Stéphane Grabli e colaboradores, cujos estilos de linha sejam programáveis. Também apresentamos uma implementação de parte do trabalho de Grabli.

Palavras chaves: Matemática, Matemática Aplicada, Computação Gráfica, Renderização, Renderização de Modelos 3D.

Abstract

This dissertation dicusses the main concepts involved in the non-photorealistic rendering techniques and proposes a general scheme to implement such techniques. The work discusses also a non-photorealistic rendering style from line drawing to 3D models presented by Stéphane Grabli et al, in such a way that the line styles are programmable. We also present an implementation from portion of the work of Grabli.

Key word: Mathematic, Applied Mathematics, Computer Graphics, Rendering, Rendering for 3D model.

Sumário

1 Introdução **5**
1.1 Objetivos 6
1.2 Aplicações 6
1.3 Estrutura da dissertação 6

2 Renderização Não-fotorrealística **8**
2.1 Renderização de imagens 8
2.1.1 Exemplos de Método de RNF de imagens 14
2.2 Renderização de Modelos 3D 15
2.2.1 Alguns Métodos de RNF de Modelos 3D 20

3 Esquema Geral para RNF **22**
3.1 Módulo Objeto 22
3.2 Módulo de Extração de Estruturas Morfológicas 23
3.3 Módulo de Sombreamento 24

4 Um Método de RNF para Desenhos de Linhas **26**
4.1 Esquema Geral do Método 26
4.2 Módulo de Extração 28
4.3 Módulo de Shader 30

5 Resultados **38**

6 Conclusões e Trabalhos Futuros **40**
6.1 Conclusões 40
6.2 Trabalhos Futuros 41

Bibliografia **44**

Lista de Figuras

2.1 O processo de esqueletonização utilizando o eixo medial. (Extraída de [24]) 10
2.2 Exemplo da utilização de um *chain code* para extração de uma curva poligonal. 10
2.3 Identificação de vetores normais do Homer Simpson. 11
2.4 Rotulação feita com o uso de componentes conexas. 12
2.5 Interpolação de normais no espaço RGB. 12
2.6 Exemplo de sombreamento *cartoon* 13
2.7 Estilo de sombreamento de*phong* 14
2.8 Exemplo de colorização. 15
2.9 (a) imagem original; (b) detecção de silhuetas; (c) segmentação de cor; regiões contínuas de cor sólida nesta imagem representam elementos individuais de segmentação; (d) segmentação de cor numa escala grosseira. 16
2.10 Arestas Silhuetas 16
2.11 Exemplo do método de Hertzman para calcular a silhueta. 17
2.12 Importância de contornos sugestivos de uma imagem. . . . 18
2.13 Arestas fronteiras de um hexágono. 18
2.14 Tratamento de arestas escondidas de um dodecaedro. . . . 19
2.15 Exemplo de uma sombreamento estruturado. Figura extraída de [3] 19
2.16 Exemplo de um sombreamento livre. Figura extraída de [3] 20
2.17 Um *Toon Shader* e um sombreamento móvel sugere uma iluminação no homem da neve. [3] 20
2.18 Do mesmo modelo de xícara de chá 3D podem ser produzidos quatro estilos de renderização distintos. 21
2.19 Exemplo de Ilustração Pen-and-Ink 21

3.1 Esquema geral para renderização não-fotorrealística. 22
3.2 Módulo Objeto 23
3.3 Módulo Extração 24
3.4 Módulo Shader 25

4.1 Arquitetura do sistema 27
4.2 Silhueta extraída do modelo 3D da deusa Atenas. 29
4.3 Contornos Sugestivos do modelo 3D. 29
4.4 Tratamento de arestas escondidas dos contornos sugestivos de um cubo. 30
4.5 Foi dada a silhueta a cor rosa e espessura 3. Já aos contornos sugestivos, foi aplicada uma cor azul e espessura 1. 30
4.6 Exemplo dos operadores propostos por Grabli onde são usados dentro dos três módulos de estilo constituindo um estilo sheet simples. 31
4.7 Exemplo de predicativos de encadeamento simples, aplicados a um conjunto de *ViewEdges* no contorno externo de um desenho. Resultados obtidos por Grabli. 33
4.8 Uso de múltiplos strokes através de cadeias para renderizar construção em um estilo esboçado. Resultado também obtido por Grabli 34
4.9 Topo: Desenho simplificado de uma área de visão complexa. Inferior esquerdo: visão close-up com todas as linhas visíveis mostrando a complexidade visual não desejada. Resultado obtido por Grabli. 36

5.1 Janela de visualização dos arquivos *ply*. 38
5.2 (Esquerda)Janela de visualização do ViewMap. (Direita) Janela onde o usuário pode modificar os atributos da silhueta e dos contornos sugestivos separadamente. 38
5.3 Aplicamos a cor amarela e espessura 2 para a silhueta e cor rosa e espessura 1.5 para os contornos sugestivos. 39
5.4 Neste exemplo damos ênfase a silhueta do objeto mudando a sua cor para azul e aumentando a espessura da linha para 2. 39
5.5 Foi dada ao contorno sugestivo do tênis uma cor vermelha e espessura 2. 39

Lista de Tabelas

1.1 Fotorrealismo X RNF 5

Capítulo 1

Introdução

Os métodos de renderização fotorrealísticos objetivam a geração de imagens indistinguíveis de uma fotografia. Em algumas situações, como cartuns, pinturas artísticas, desenhos técnicos e visualização científica, realismo não é prioridade. Por isso, a renderização não-fotorrealística (RNF), também chamada renderização estilizada, possui uma grande importância. Por se concentrar apenas nos detalhes de maior interesse, a RNF apresenta algumas vantagens, tais como uma melhor visualização e entendimento das imagens, um menor tempo de renderização, um menor espaço de armazenamento, além da possibilidade de expressões artísticas.

Na tabela 1.1 (ver [2]), temos uma comparação entre a renderização fotorrealística e a RNF.

	fotorrealismo	RNF
Aproximação	Simulação	Estilização
Característica	Objetiva	Subjetiva
Influências	Simulação de processos físicos	Afinidade com processos artísticos; baseado em percepção
Precisão	Precisa	Aproximada
Engano	Pode ser desonesto	Honesto
Nível de detalhes	Nível constante de detalhes	Pode adaptar o nível de detalhe transversalmente uma imagem para focar a atenção do observador.
Bom para detalhar	Superfícies Rígidas	Fenômeno natural e orgânico

Tabela 1.1: Fotorrealismo X RNF

Em síntese, a renderização não-fotorrealística produz imagens mais significativas omitindo detalhes desnecessários, focalizando a atenção em características relevantes, esclarecendo aspectos e revelando partes escondidas. Além disso, ela também é um veículo natural para exibir informações em vários níveis de detalhes.

1.1 Objetivos

Este trabalho tem como principais objetivos:

- discutir os principais conceitos envolvidos nas técnicas de RNF,
- sugerir um esquema geral para implementação de tais técnicas, e
- discutir um estilo de renderização não-fotorrealística de desenhos de linha para modelos 3D apresentado por [1], cujos estilos de linha sejam programáveis.

1.2 Aplicações

Estilos de RNF são usados em diferentes contextos como ilustrações técnicas e científicas, manuais de ferramentas, mapas, símbolos e artes.

Em projetos arquitetônicos, por exemplo, um arquiteto apresenta ao cliente um projeto preliminar para uma casa ainda não construída. A primeira versão é um esboço impreciso, feito à lápis, que omite muitos detalhes sugestivos para o cliente, ficando em aberto para revisão, concedendo a troca de idéias entre o cliente e o arquiteto.

Outro exemplo do uso de desenhos em lugar de fotografias é o caso de livros da área médica. Eles empregam desenhos de linhas esquemáticas para ilustrar estruturas da anatomia. Isto porque fotografias tendem a implicar uma precisão e perfeição da cena para o objeto real. Ilustrações desenhadas à mão podem melhorar a comunicação da estrutura 3D, omitir os detalhes desprezíveis e enfatizar somente características importantes.

A criação de imagens em ilustrações técnicas é outro exemplo. Somente aqueles detalhes relevantes para o objetivo de uma aplicação particular são mostrados. Por exemplo, uma fotografia reluzente de um motor pode ser útil para vender o carro para um cliente, mas para consertar o motor, um desenho de linha simplificado com as partes relevantes destacadas pode ser significante.

1.3 Estrutura da dissertação

- No capítulo 2 falaremos um pouco mais sobre renderização não-fotorrealística (RNF), diferença entre renderização de imagens e renderização de modelos 3D e daremos alguns exemplos de cada uma dessas renderizações.

- No capítulo 3 exibiremos um esquema geral para RNF.
- No capítulo 4 trataremos do método descrito por Grabli e colaboradores [1] sobre uma técnica programável para desenhos de linha não-fotorrealísticos. Mostraremos um cálculo para silhuetas e contornos sugestivos, que é a parte inicial e a implementação do método descrito acima.
- Já no capítulo 5, exibiremos alguns resultados obtidos da nossa implementação.

Capítulo 2

Renderização Não-fotorrealística

Este capítulo discute os principais elementos e conceitos envolvidos nos métodos de renderização não-fotorrealística e faz um levantamento das principais técnicas.

Como dito anteriormente, a RNF tem se tornado uma importante área de pesquisa em computação gráfica. Um fato interessante nesta área é que muitas das técnicas desenvolvidas durante anos de pesquisa em computação gráfica podem ser usadas para criar efeitos específicos. A renderização não-fotorrealística é uma técnica que produz imagens que simulam o mundo 3D em um estilo diferente do realismo. Renderização não-fotorrealística é uma representação de estilos para imagens de computador que usam técnicas para criações artísticas, pinturas ou cenas 3D cujos efeitos podem ter a aparência de um desenho animado tradicional. O resultado deve transmitir a idéia de uma ilustração artística feita à mão.

Dentro das diversas técnicas de RNF vamos destacar dois tipos de renderização: a renderização não-fotorrealística de imagens e a renderização não-fotorrealística de modelos 3D. A renderização de imagens podem ter o intuito de, dada uma imagem, aplicar efeitos que façam com que essa imagem tenha aspectos visuais 3D ou que simplesmente modifiquem sua aparência. Já a renderização 3D toma como entrada um objeto 3D e tenta transformá-lo em uma imagem 2D, extraindo linhas características do objeto 3D, como silhuetas, contornos sugestivos, dentre outros. As próximas seções destacam esses dois tipos de RNF.

2.1 Renderização de imagens

Estas técnicas são aplicadas a imagens com o objetivo de mudar o aspecto visual dos objetos presentes nas imagens. Alguns métodos recorrem a ferramentas que reconstroem ou simulam uma geometria 3D dos objetos e

em seguida aplicam técnicas de visualização tradicionais para dar à imagem o efeito desejado como, por exemplo, [10]. Outros métodos simplesmente aplicam algoritmos de processamento de imagens e visão computacional para identificar os elementos da imagem e atribuir os mais variados efeitos como [14]. Estas duas técnicas também podem ser combinanadas para a definição de outros métodos.

Uma aplicação de renderização não fotorrealística de imagens é a iluminação de cartoons. Nesta aplicação um cartoon ou desenho feito à mão é digitalizado. Em seguida esta imagem é processada de modo que o resultado final seja um desenho colorido, iluminado com efeitos 3D. O cálculo da iluminação requer que o mapa de normais 3D do desenho seja conhecido e, desse modo seja possível computar a interação das superfícies do desenho com as luzes presentes em um ambiente 3D. Porém nos cartoons a normal da superficie é desconhecida e a informação de posição carece de profundidade. Por isso, são necessário alguns processos para extrair essas informações para que se possa, por exemplo, inferir geometrias tridimensionais nessas imagens aplicando técnicas de sombreamento. Estes processos possibilitam a geração de efeitos visuais 3D na imagem, sem a necessidade de se calcular a tridimensionalidade do modelo. Mostraremos em seguida alguns dos processos utilizados na renderização de imagens.

Esqueletonização *Pixels* sobressalentes podem complicar a análise da imagem, e portanto devem ser removidos. A esqueletonização calcula um conjunto linear dos contornos dos objetos da imagem. Este conjunto é espacialmente colocado ao longo do *eixo medial*, que é o conjunto de pontos internos ao objeto de modo que sejam eqüidistantes da borda deste objeto (veja [24]). O eixo medial, assim como outros tipos de esqueletos, contém propriedades topológicas e geométricas do objeto e reproduz sua forma global. Assim, algoritmos de esqueletonização "corroem" a imagem reduzindo objetos sem mudar sua topologia. Na figura 2.1 vemos um exemplo de esqueletonização em uma imagem 2D.

Existem na literatura diversos outros métodos que tratam da esqueletonização de imagens ([13, 25, 26]).

Extração das curvas Para conseguir discernir os objetos de uma imagem, é necessário detectar os contornos dos objetos, denominado de arestas. Esse processo extrai curvas poligonais que representam o contorno dos objetos presentes na imagem. Os algoritmos de extração de curvas obtêm informações sobre a estrutura morfológica dos objetos

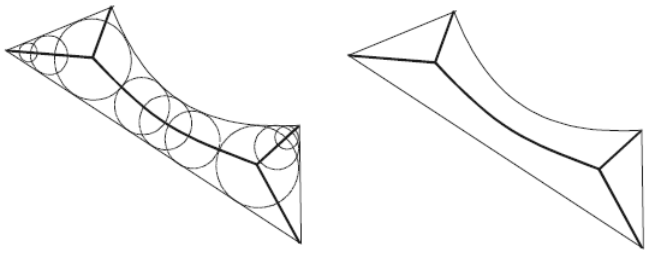

Figura 2.1: O processo de esqueletonização utilizando o eixo medial. (Extraída de [24])

da cena. Uma das técnicas mais utilizadas para extração das curvas é o *chain code* [20]. Os chain codes são usados para representar as fronteiras de regiões através de segmentos de retas de tamanho e direção específicas. Normalmente, esta representação tanto pode usar a vizinhança 4-conectado ou 8-conectado e a direção de cada segmento é codificada usando um esquema de numeração chamado código de Freeman. Imagens digitais são adquiridas e processadas em um formato de grade com espaçamento uniforme nas direções x e y. Desta forma um *chain code* pode ser gerado pela perseguição no sentido horário dos segmentos que ligam cada par de pixels e atribuindo o código de direção a este segmento e transformando a curva suave em uma curva poligonal.

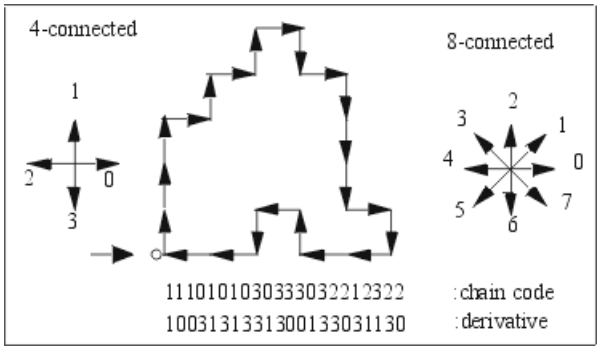

Figura 2.2: Exemplo da utilização de um *chain code* para extração de uma curva poligonal.

Cálculo das normais as curvas Outro processo importante é o calculo das normais as curvas dos objetos da imagem. De posse da estrutura morfológica de uma curva poligonal, o vetor normal em cada ponto é facilmente determinado tomando-se o vetor perpendicular ao vetor direção da curva. Figura 2.3.

Pode ser feita também uma suavização dessas normais já que, na maioria das vezes, existem apenas oito direções possíveis de vetores normais em cada ponto da curva. Faz-se então necessário uma leve "perturbação" na direção desses vetores a fim de obter-mos mais direções

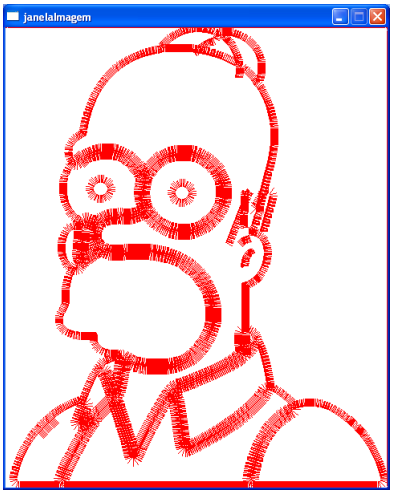

Figura 2.3: Identificação de vetores normais do Homer Simpson.

para cada ponto. Uma forma de fazer essa suavização é: para cada ponto, o vetor normal em cada ponto é calculado pela adição dos vetores normais de seus pontos vizinho anterior e posterior.

Rotulação A rotulação faz a identificação de regiões na imagem não-sobrepostas, cada uma sendo significante de acordo com uma aplicação particular. Um algoritmo que pode ser usado é *flood fill* [21] que identifica todos os pixels contidos na região interior a essa curva. Também pode ser usado para a rotulação o conceito de *componentes conexas*. De acordo com [27], Componente Conexa é um conjunto finito de pixels conectados que compartilham uma determinada propriedade V. Dois pixels p e q são conectados ou conexos quando existe um caminho de p até q onde todos os pixels do caminho apresentam a propriedade V. A rotulação por componentes conexas faz com que todos os pixels em uma componente conexa possuam o mesmo rótulo e faz com que todas componentes diferentes possuam rótulos distintos. A figura 2.4, mostra o exemplo de uma rotulação por componente conexa feita no modelo do Homer Simpson.

Interpolação Uma vez que temos os vetores normais sobre a curva, o estágio de interpolação aplica interpolação esparsa para aproximá-los sobre a imagem restante. As componentes nx, ny e nz são calculadas em cada ponto da imagem a partir das normais das curvas. Dois métodos de interpolação possíveis são [28]: trocar cada vetor normal por outro de uma esfera escalada em z ou trocar cada vetor normal por outro de uma porção de uma esfera escalada uniformemente. A figura 2.5 mostra um exemplo de uma interpolação com as normais

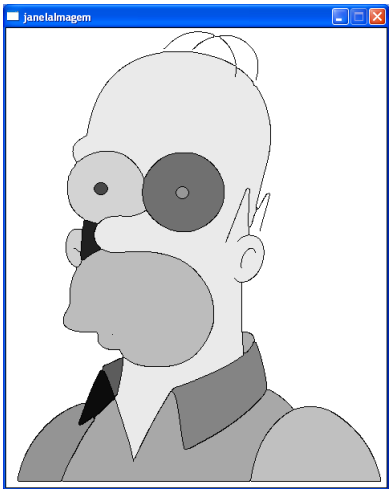

Figura 2.4: Rotulação feita com o uso de componentes conexas.

no espaço RGB.

Figura 2.5: Interpolação de normais no espaço RGB.

Sombreamento Outra ferramenta importantes na renderização de imagens é o sombreamento. Vamos falar brevemente sobre dois métodos de sombreamento: o de *Phong* e o *Cartoon*. Sombreamento é um método fundamental para a ilusão de uma estrutura 3D de um objeto em uma imagem bidimensional. Pesquisadores em renderização não-fotorrealística têm investigado uma variedade de técnicas que simulam sombreamentos encontrados na arte. No sombreamento *Cartoon* sombreia-se a parte do material que está na sombra com uma cor que é mais escura do que a cor do material. Esse truque adiciona idéia de iluminação, de forma e de contexto do personagem na cena. A Figura 2.6 mostra um exemplo de sombreamento *cartoon*.

A equação de iluminação utilizada para calcular a iluminação di-

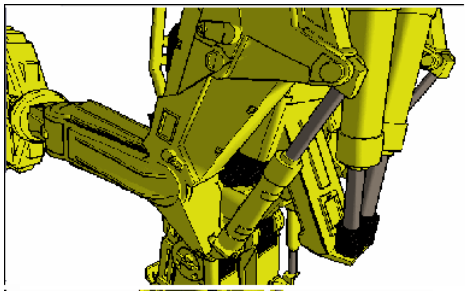

Figura 2.6: Exemplo de sombreamento *cartoon*

fusa nos vértices para ambos os sombreamento suave e sombreamento Gouraud é descrita como:

$$C_i = a_g \times a_m + a_l \times a_m + (Max\{\bar{L} \cdot \bar{n}, 0\}) \times d_l \times d_m \tag{2.1}$$

onde C_i é a cor do vértice, a_g é o coeficiente global da luz ambiente, a_l é o coeficientes ambiente e difuso para a fonte de luz, a_m é o coeficiente ambiente e difuso para material do objeto, $\bar{L}$ é o vetor unitário a partir da fonte de luz até o vértice, n é o vetor unitário normal à superfície do vértice e $\bar{L} \cdot \bar{n}$ é o cosseno do ângulo entre os dois vetores.

Já o sombreamento de Phong é provavelmente o mais utilizado em computação gráfica hoje em dia. Nesse modelo, a luz em qualquer ponto é composta por três componentes: ambiente, difusa e especular. Essas três componentes são aditivas e determinam o aspecto final da iluminação e da cor de um determinado ponto na cena ou da superfície de um determinado polígono plano contido nela. Através dos vetores normais obtidos através do processamento descrito acima, o modelo de iluminação de Phong pode ser aplicado na imagem uma vez que as intensidades das fontes de luz (ambiente, difusa e especular), seus coeficientes (difuso e especular) e as posições do observador e da fonte de luz sejam definidos pelo animador. Finalmente, o modelo de Phong pode ser aplicado à imagem através de sua equação:

$$I = I_{amb} + I_{dif} + I_{espec} \tag{2.2}$$

onde:

$$\begin{aligned} I_{amb} &= I_A r_{amb} \\ I_{dif} &= I_L r_{dif}(\vec{v}_L \cdots \vec{v}_n) \\ I_{espec} &= I_L r_{espec}(\vec{v}_{ref} \cdots \vec{v}_{obs})^s \end{aligned}$$

sendo I_{amb} a intensidade da luz ambiente da superfície, I_A a intensidade global da luz ambiente, r_{amb} o coeficiente de refletividade ambiente da superfície, I_dif a intensidade da luz difusa, I_L a intensidade da fonte de luz, r_{dif} o coeficiente de refletividade difusa da superfície.$\vec{v}_L$ o vetor normalizado apontando para a fonte de luz, $\vec{v}_n$ o vetor da superfície, I_{espec} a intensidade de a luz especular, I_L a intensidade da fonte de luz, r_{espec} o coeficiente de refletividade especular da superfície. $\vec{v}_{ref}$ o vetor normalizado apontando para a direção de reflexão. $\vec{v}_{obs}$ o vetor normalizado apontando para a direção do observador e s o expoente especular. Na figura 2.7 vemos um exemplo de sombreamento em estilo Phong.

Figura 2.7: Estilo de sombreamento de*phong*

A utilização do modelo de *Phong* possibilita resultados de grande apelo tridimensional, aproximando com bastante eficácia a iluminação incidente sobre superfícies curvas. Se acrescidos de detalhes como silhuetas e contornos, os resultados obtidos perdem um pouco do aspecto artificial comumente encontrado nas imagens geradas por computador.

Na subseção 2.1.1 serão exibidos dois exemplos de RNF de imagens.

2.1.1 Exemplos de Método de RNF de imagens

Um trabalho recente nesta área é a colorização 3D para animação 2D [10] que tem como principal tarefa prover um conjunto de técnicas para permitir a colorização semi-automática em estilos 2D/3D de uma seqüência

de quadros numa animação bidimensional combinando métodos da computação gráfica e processamento de imagens. Essa técnica permite que imagens bidimensionais possam ser coloridas em estilos tridimensionais utilizando algoritmos de processamento de imagens semi-automáticos. Essa aplicação permite pré-processar imagens 2D a fim de obter mapas de normais que aproximem a geometria do desenho. Após este processo, a colorização tridimensional é provida através de técnicas de sombreamento pré-implementadas como *phong*, *cartoon* e mapeamentos de reflexão. Um algoritmo de rastreamento é disponibilizado como forma de automatizar a colorização de cada quadro da animação. A figura 2.8, mostra um exemplo de colorização em um urso.

Figura 2.8: Exemplo de colorização.

Outro exemplo de renderização de imagens é o trabalho de DeCarlo e Santela [14]. Eles descrevem uma aproximação para estilização e abstração de fotografias que responde em termos explícitos ao objetivo do projeto de esclarecer a estrutura visual significante em uma imagem. A aproximação deles renderiza imagens em um estilo de desenho de linha usando arestas irrelevantes e regiões grandes de cor constante. Para isto, ele representa imagens como uma estrutura hierárquica de partes e fronteiras calculadas usando visão computacional. Ele identifica os elementos significantes dessa estrutura usando um modelo de percepção humana. O sistema renderiza uma nova imagem usando transformações que preservam e destacam esses elementos visuais. Um exemplo desse método de renderização pode ser visto na figura 2.9.

2.2 Renderização de Modelos 3D

Estas técnicas de RNF recebem com entrada uma malha (em geral triangular) e geram uma imagem a partir da extração de linhas características e outras informações do modelo 3D. Discutiremos a seguir alguns passos para extração dessas linhas e outras informações importantes.

Silhuetas Silhueta é o conjunto de arestas que formam o contorno de um

Figura 2.9: (a) imagem original; (b) detecção de silhuetas; (c) segmentação de cor; regiões contínuas de cor sólida nesta imagem representam elementos individuais de segmentação; (d) segmentação de cor numa escala grosseira.

objeto para certo observador. Ela delimita os modelos evidenciando os que estão em primeiro plano e relaciona o objeto com o mundo. Matematicamente, a silhueta é caracterizada pelos pontos de uma superfície projetados na imagem, onde o vetor de visão é paralelo ao plano tangente da superfície. Em malhas poligonais, consiste de todas as arestas que conectam faces poligonais escondidas de faces poligonais visíveis. Já em superfícies suaves, a silhueta pode ser definida como o conjunto dos ponto p da superfície com uma normal à superfície $n(p)$ perpendicular ao vetor de visão:

$$n(p) \cdot (p - \mathbf{C}) = 0 \tag{2.3}$$

onde $\mathbf{C}$ é o centro da câmera.

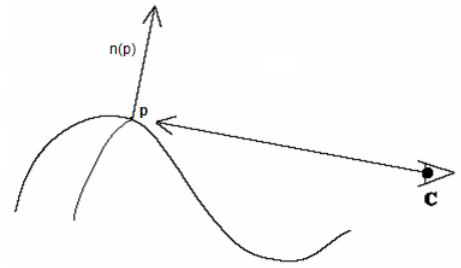

Figura 2.10: Arestas Silhuetas

Note que esta definição não cria lembrança de visibilidade; um ponto que é obstruído por outro objeto pode ser agora ponto de silhueta. Em quase todos os casos, nós estamos somente interessados em renderizar o segmentos visíveis de curvas silhuetas, ou em renderizar as partes invisíveis como uma linha característica diferente. Existem vários métodos de calcular silhuetas. O artigo de Hertzman e Zorin [5] nos ensina um método para calcularmos tais arestas silhuetas. Eles definiram uma função $g(p) = (n(p) \cdot (\mathbf{p} - \mathbf{c}))$ que é calculada para todos os pontos da aproximação poligonal. É calculado então o sinal

da função $g(p)$ em cada vértice. Se dois vértices adjacentes a mesma aresta possuem sinais trocados quando calculada essa função, podemos afirmar que a silhueta passa entre esses dois vértices (pela definição de silhueta dada acima). Então é feita uma interpolação entre esses dois vértices para saber onde está o ponto em que passa a silhueta. Na figura 2.11 podemos ter uma idéia desse método.

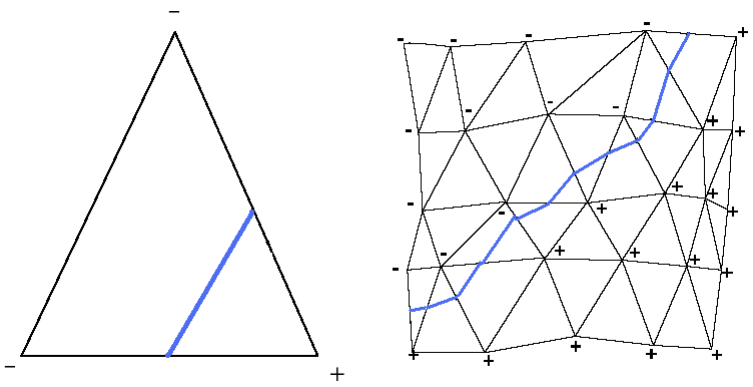

Figura 2.11: Exemplo do método de Hertzman para calcular a silhueta.

Outra forma de calcular a silhueta é usando intuitivamente a definição dada para malhas poligonais, que a silhueta é o conjunto de todas as arestas que conectam faces poligonais escondidas de faces poligonais visíveis. Para isso, calcula-se o produto escalar entre o vetor de visão menos o ponto médio entre o vértice inicial e final da aresta e a normal de cada face. Se nas faces adjacentes a mesma aresta o produto escalar for maior que zero em uma e menor que zero na outra, então esta aresta é uma silhueta. Ou seja, suponha uma aresta e de uma determinada malha, n_1 e n_2 as normais das faces f_1 e f_2 adjacentes respectivamente e m o ponto médio entre o vértice inicial e final da aresta. Seja $P_1 = \langle n_1, C - m\rangle$ e $P_2 = \langle n_2, C - m\rangle$, se $P_1 > 0$ e $P_2 < 0$ ou $P_1 < 0$ e $P_2 > 0$, então e é uma aresta silhueta, onde C é o vetor de visão.

Contornos sugestivos DeCarlo e colaboradores [4] definem informalmente os contornos sugestivos como curvas cuja *curvatura radial* é zero e onde a superfície curva-se para fora do observador. A curvatura radial é a curvatura normal da superfície na direção **w** definida como a projeção (não normalizada) do vetor de visão $v(p)$ no plano tangente em **p**. Equivalentemente, um ponto p faz parte de um contorno sugestivo se:

$$0 < \theta_c < \cos^{-1}\left(\frac{n(p).v(p)}{\|v(p)\|}\right)$$

Na figura 2.12 (extraída de [4]) vemos um exemplo mostrando a expressividade adicionada pelos contornos sugestivos. A imagem da esquerda é desenhada usando somente os contornos, enquanto a imagem da esquerda usa contornos e contornos sugestivos.

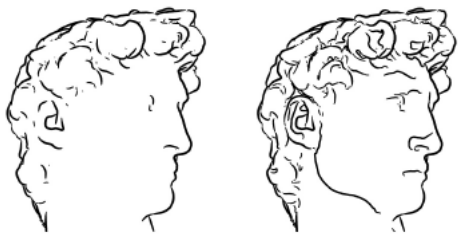

Figura 2.12: Importância de contornos sugestivos de uma imagem.

Outra forma de encontrar essas arestas escondidas é calcular o produto interno entre as duas normais das faces adjacentes de cada aresta, se esse produto interno for menor a um valor entre 0 e 1 predeterminado, dependendo do nível de detalhes, essa aresta é um contorno sugestivo.

Fronteira É uma aresta que não é comum a dois polígonos, tal como a aresta de uma folha de papel. Um objeto sólido tipicamente não tem arestas de fronteira. Na figura abaixo foi extraídas essas arestas de um modelo 2D de um hexágono.

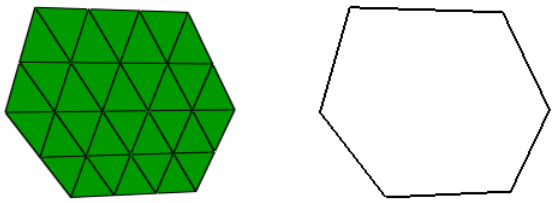

Figura 2.13: Arestas fronteiras de um hexágono.

Strokes Coleção de diferentes tipos de traços, cada um com uma prioridade diferente . Os strokes também podem armazenar os atributos para cada tipo de traço. Muitas da técnicas de RNF calculam as linhas características do modelo e em seguida as armazenam nessa estrutura. Ver [22]

Eliminação de arestas escondidas Na extração de algumas linhas podem estar inclusas faces e arestas irrelevantes. Uma face é relevante se o produto escalar entre sua normal e a direção de visão é positivo. Uma aresta é relevante se ela é a interseção de duas faces relevantes.

Podemos então excluir essas linhas dependendo da utilização. Existe um conceito que nos ajuda a calcular tais arestas escondidas que é a *invisibilidade quantitativa* (ver [6]). Um ponto p tem invisibilidade quantitativa n se existem n polígonos entre o observador e p. Como conseqüência imediata dessa definição, um ponto p é visível se, e somente se, a sua invisibilidade quantitativa é zero. Appel [6] define uma maneira de calcular arestas escondidas utilizando invisibilidade quantitativa. Na figura 4.4 vemos na esquerda as linhas características de um dodecaedro sem tratamento de faces escondidas e na direita o mesmo dodecaedro agora excluindo faces escondidas.

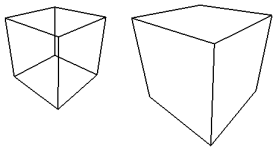

Figura 2.14: Tratamento de arestas escondidas de um dodecaedro.

Uma forma intuitiva para calcular arestas escondidas é: dada as normais n_1 e n_2 das faces f_1 e f_2 adjacentes a uma aresta e respectivamente, seja $u_1 = \langle C, n_1 \rangle$ e $u_2 = \langle C, n_2 \rangle$, onde C é a direção da visão, se $u_1 < 0$ e $u_2 < 0$ então e é uma aresta escondida, caso contrário, e é uma aresta visível.

Sombreamento Outro etapa importante é o sombreamento tal como estruturado, livre, e móvel. Vamos descrever alguns detalhes de cada um deles. Sobre os sombreamentos citados aqui, podemos obter mais informações em [3]. O *Sombreamento Estruturado* é um tipo de sombreamento que utiliza linhas paralelas. O tamanho e quantidade de linhas variam com a distância do objeto à câmera. A figura 2.15 mostra primeiro a mudança do nível de detalhes a medida que a câmera aproxima em um grupo de *strokes*.

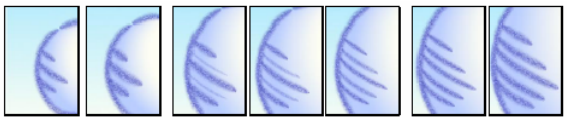

Figura 2.15: Exemplo de uma sombreamento estruturado. Figura extraída de [3]

Já o *Sombreamento livre*, como o próprio nome já diz, não existe padrão no sombreamento e é necessário que seja feito para várias distâncias diferentes do objeto. Por exemplo, a figura 2.16 mostra um exemplo de sombreamento livre. O *Toon shading*, o qual é translúcido

nas áreas escuras e inteiramente transparente nas áreas iluminadas, lembra uma aquarela. Níveis múltiplos de detalhes são mostrados no sombreamento livre quando o zoom da câmera em relação a uma área de visão fechada (direita).

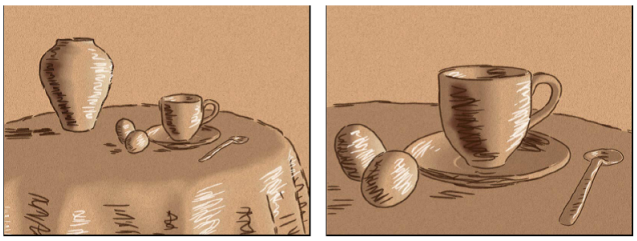

Figura 2.16: Exemplo de um sombreamento livre. Figura extraída de [3]

O *Sombreamento Móvel* É um tipo de sombreamento que se move na figura de acordo com a posição da luz. Utiliza-se um modelo de "luz escura" para calcular o sombreamento. Esse efeito é usado, por exemplo, no homem de neve na figura 2.17.

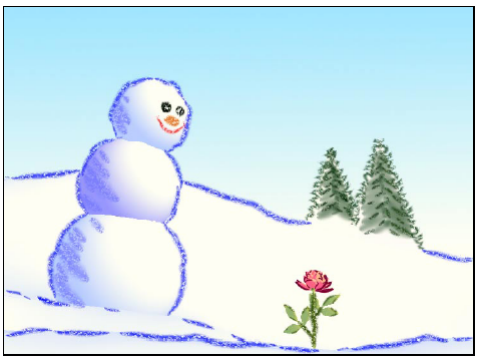

Figura 2.17: Um *Toon Shader* e um sombreamento móvel sugere uma iluminação no homem da neve. [3]

2.2.1 Alguns Métodos de RNF de Modelos 3D

Um exemplo de renderização 3D é o WYSIWYG NPR [3]. Ele faz uma estilização automática de um modelo de uma malha triangular a partir de alguns traços de exemplo. Ele foca-se em algoritmos de renderização baseados em *strokes*, com três categorias principais de *strokes*: silhuetas e linhas de dobras que forma a base dos desenhos de linha simples; *decal strokes* que sugere características da superfície; e *hatching strokes* para conduzir iluminação artificial e tonalidade. Em cada caso, é fornecida uma interface para direcionar o controle do usuário, e algoritmos de renderização em tempo real para sustentar a interatividade requerida. O designer pode

aplicar *strokes* em cada categoria com variação estilística significante, e assim em combinação executar uma escala ampla de efeitos. Um exemplo desse método de renderização é visto na figura 2.18, [3].

Figura 2.18: Do mesmo modelo de xícara de chá 3D podem ser produzidos quatro estilos de renderização distintos.

Outro exemplo é a Ilustração *Pen-and-Ink* [12]. Os desenhos têm as seguintes propriedades: a pena não tem variação de cor ou tonalidade, todo sombreamento deve ser feito utilizando traços; é manualmente difícil e trabalhoso preencher grandes áreas; é ideal para contornos, cada traço pode ter variações na pressão da pena e irregularidades do traçado,e; fornece-nos imagens simples e diretas. A ilustração Pen-and-Ink nos fornece dualidade de traços: normalmente a renderização de textura e tonalidade são feitas separadamente e neste caso é preciso que o mesmo traço seja utilizado para textura e tonalidade. Ela combina informações 2D e 3D pois normalmente as informações utilizadas para renderizar são 3D e depois projetadas em 2D e, por isso, as ilustrações pen-and-ink as informações da projeção 2D são tão importantes quanto as informações em 3D pois precisamos considerar as áreas das projeções para calcular a densidade dos traços e usam-se as adjacências em 2D para criar os contornos. Na figura 2.19 vemos um exemplo dessa técnica.

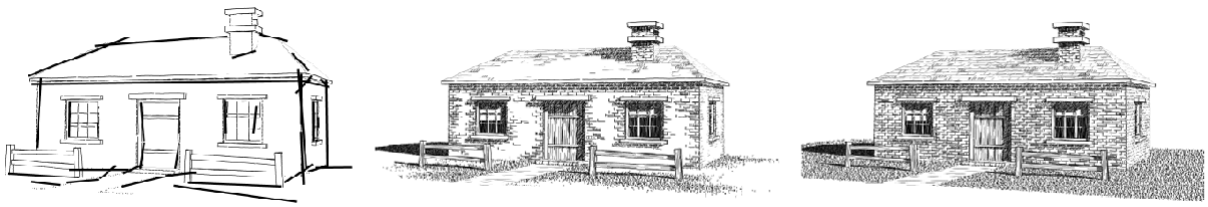

Figura 2.19: Exemplo de Ilustração Pen-and-Ink

Capítulo 3

Esquema Geral para RNF

Neste capítulo vamos apresentar um esquema geral que permite o desenvolvimento e implementação de métodos de renderização não-fotorrealística. De forma geral, um método de renderização não-fotorrealística recebe como entrada um objeto (uma imagem, um modelo 3D, etc.) e procura extrair conjuntos de estruturas que representem a topologia, geometria, formas, regiões e contornos representativos do objeto. Aliás, a visão humana utiliza este tipo de estratégia para capturar a forma geral dos objetos que nos rodeiam. A partir dessas estruturas extraídas, os métodos de RNF aplicam processos de renderização, aplicando efeitos ao objeto de acordo com o estilo desejado. Como saída, temos uma imagem estilizada. Na figura 3.1 podemos observar os passos para renderização, onde cada um desses passos podem ser representados como uma classe no processo de implementação da RNF. Nas próximas seções, explicaremos passo a passo a idéia de cada uma dessas classes.

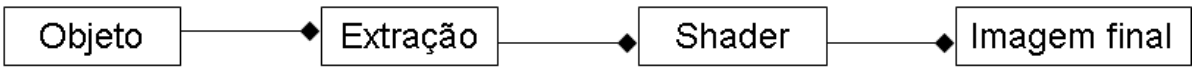

Figura 3.1: Esquema geral para renderização não-fotorrealística.

Em [1] é apresentado um esquema para implementação de métodos de RNF. Porém o esquema se restringe à renderização apenas das linhas de módulos representados por malhas triangulares.

3.1 Módulo Objeto

Vamos considerar a renderização de imagens e de modelos 3D. Toma-se como entrada um modelo 3D ou uma imagem. Uma imagem pode ser representada, em duas dimensões, como um conjunto finito de valores digitais, chamados pixels. A matriz é uma malha, onde cada ponto ou célula

é um pixel, com um valor associado a cada ponto. Esse valor é chamado de intensidade da imagem e representa alguma propriedade, como cor, tonalidade, brilho e outras, ou seja, é como se tivéssemos uma tabela de correspondência do número às várias cores. Uma das formas de representação da cor é por combinação linear de três cores: vermelho, verde e azul, o RGB.

Os modelos 3D podem ser representados por uma malha $M = \{V, A, F\}$, onde V é um conjunto de vértices, A é um conjunto de arestas e F é um conjunto de faces triangulares. Podem ser armazenados como arquivos, tal como arquivos *ply*, que é um formato de arquivo de computador conhecido por *Polygon File Format* ou *Stanford Triangle Format*. Esse formato foi principalmente concebido para armazenar dados tridimensional a partir de *scanners* 3D. A figura 3.2 mostra esses dois tipos de objetos que podem ser tratados pelo nosso esquema.

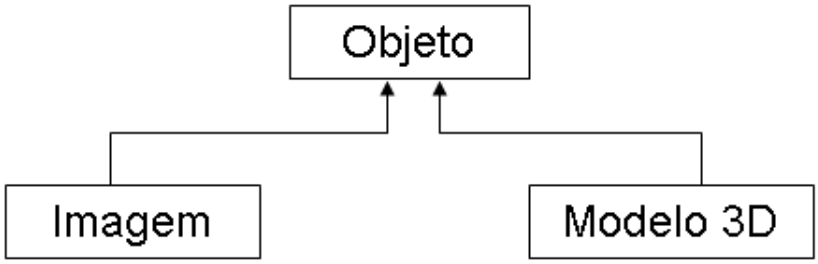

Figura 3.2: Módulo Objeto

3.2 Módulo de Extração de Estruturas Morfológicas

Os objetos a serem renderizados possuem estruturas morfológicas importantes que apresentam tanto seu aspecto global, quanto partes específicas que possuem determinadas características. Uma imagem, por exemplo, possui estruturas como esqueleto e seqüência de *pixels* que caracterizam informações importantes dos objetos presentes na imagem. Já um modelo 3D possui em sua silhueta contornos globais sobre o contorno do objeto. Isso sugere a definição de um módulo chamado Extração (figura 3.3) que tem como entrada um objeto (uma imagem ou um modelo 3D) e nos retorna informações da forma do objeto, como curvas esqueletos, regiões, silhuetas, etc. Este módulo pode ser visto como uma classe abstrata que possui duas subclasses especializadas: Extração de imagens e Extração de modelos. A cada uma dessas subclasses estão associadas outros módulos para extração de estruturas auxiliares.

Ao módulo de extração de imagens podem ser associados os módulos de esqueletonização, extração de curvas, cálculo de normais, etc. Dentro de cada um desses módulos podem ser implementados diversos métodos, conforme descritos na seção 2.1.

Ao módulo de Extração de Modelos 3D podem ser associados diversos módulos como silhuetas e contornos sugestivos, já definidos na seção 2.2.

Assim sendo, em qualquer caso, o módulo Extração retorna um conjunto de estruturas como curvas, regiões, normais, etc. que serão utilizados pelo próximo módulo: o *Shader*.

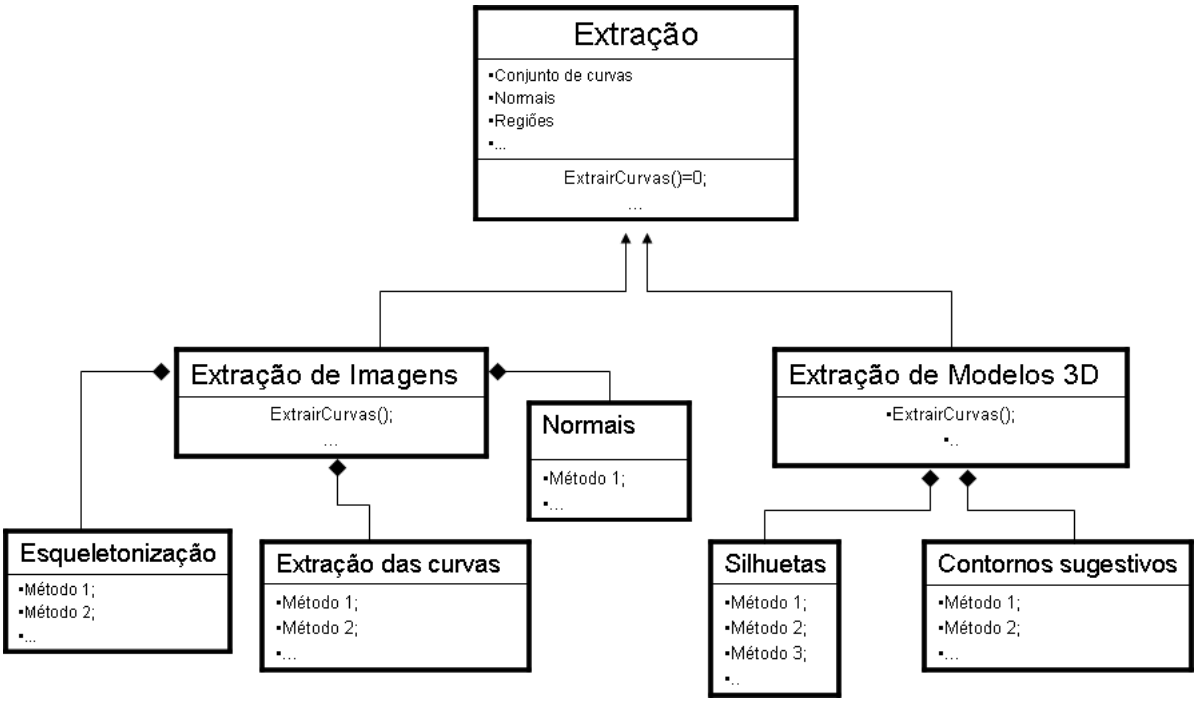

Figura 3.3: Módulo Extração

3.3 Módulo de Sombreamento

Este módulo aplica o processo de interpolação de cor para gerar uma imagem final, resultado do processo de renderização.

A partir de diversas técnicas de renderização não-fotorrealística, definimos dois sub-módulos a partir do módulo principal: O *Shader* de Contornos e o *Shader* de Área. O primeiro visa estilizar os contornos dos objetos a partir do processo de interpolação de vários atributos como espessura do traço, cor e transparência. O segundo visa criar renderização não-fotorrealística através de interpolação de cor em regiões do objeto.

O módulo de Shader pode ser projetado como uma classe abstrata que possui duas subclasses: Shader de Contornos e Shader de Áreas; o que

define praticamente cada sub-módulo é a forma como a interpolação é feita nele podem ser interpolados os métodos de renderização não-fotorrealística.

Além disso, associado ao módulo de shader há outros módulos como o de colorização que contém os módulos de iluminação que devem ser usados no cálculo de Colorização, como o modelo de Phong. Outro módulo associado ao Shader é o de Mapeamento que contém as técnicas de mapeamento de textura que podem ser aplicados aos objetos. Na figura 3.4 temos um diagrama explicativo para esse módulo.

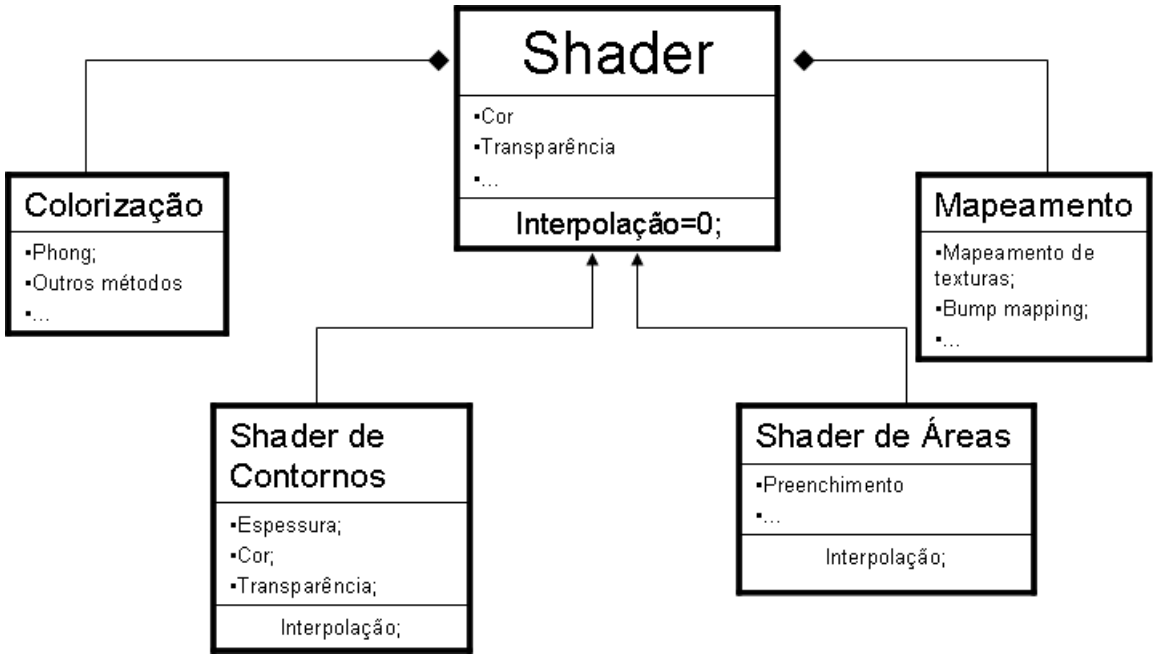

Figura 3.4: Módulo Shader

Capítulo 4

Um Método de RNF para Desenhos de Linhas

Este capítulo traz a implementação de um método de RNF as linhas ou contornos de modelos 3D. O método foi baseado no trabalho de Stéphane Grabli, e colaboradores [1] e gera contornos com linhas estilizadas e programáveis.

Desenhos de linha podem evitar desordem, focar atenção em partes relevantes e omitir detalhes que não são interessantes. O campo de renderização não-fotorrealística tem proposto uma variedade de técnicas para criação de desenhos de linhas convincentes para modelo 3D. Por outro lado, a linguagem de *shading* disponível em renderização fotorrealística tal como Pixar Renderman permite o design de uma variedade infinita de aspecto rico e complexo.

O trabalho de Grabli é baseado na hipótese que desenhos de linhas são processos seqüenciais onde são baseados em informações permitidas pela cena e o desenho atual. Essas decisões são freqüentemente explícitas, tal como ilustrações técnicas, onde densidade precisa ou atributos de cor são aplicados para diferentes linhas características [9]. Eles podem também serem implícitos, e um objetivo da pesquisa dos autores é transformá-los em explícitos e embuti-los no programa.

Nas próximas seções discutiremos como esse método pode ser visto dentro do esquema proposto no capítulo 3.

4.1 Esquema Geral do Método

Foi proposto por Grabli e colaboradores um esquema que está resumida na figura 4.1. Ela mostra a arquitetura do método dentro do esquema apresentado no capítulo anterior. Mas, como dito anteriormente, esse esquema se restringe a renderização de linhas. A entrada é um modelo poligonal 3D

e tem como saída uma imagem 2D com suas linhas renderizadas.

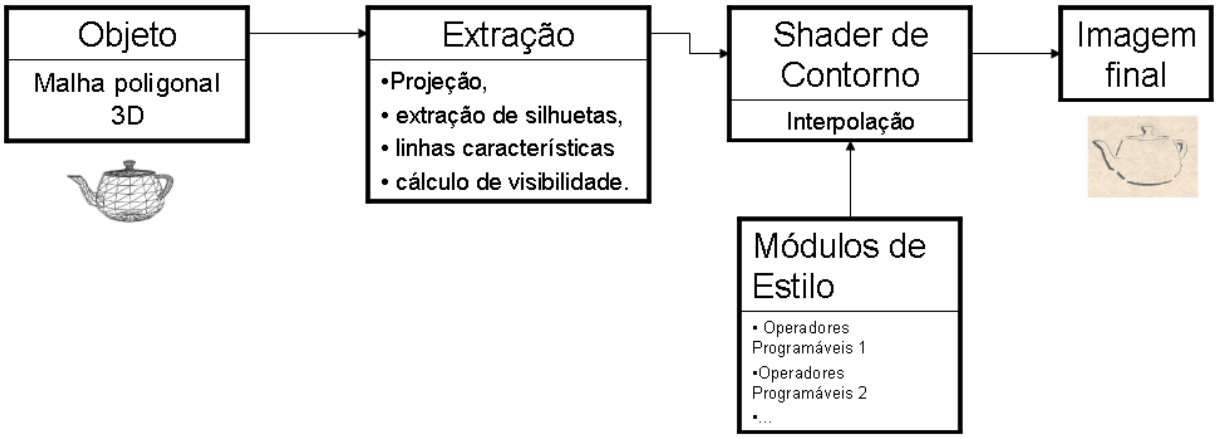

Figura 4.1: Arquitetura do sistema

Na etapa de extração é feito o cálculo de uma área de visão: um arranjo de curvas em uma imagem plana, as quais sustentarão todos os elementos do desenho. Essa implementação usa silhuetas e linhas características tal como rugas ou contornos sugestivos [5, 4, 8]. Essas imagens são projetadas em uma imagem plana, usando um algoritmo computacional de visibilidade aceitável, e elas são organizadas em um grafo planar. Um objetivo importante é guardar flexibilidade máxima na definição de marca do desenho, dessa forma o grafo deve codificar dados topológicos completos como interseções das linhas e informações de vizinhança. Isto fornece um conjunto de curvas projetadas, chamado de *ViewMap*, que compõem nossas principais estruturas: silhuetas, contornos sugestivos, etc. Também pode ser fornecido o estado da visibilidade das curvas. O conjunto de arestas de todas as curvas é chamado de *ViewEdges* e seus vértices são chamados *ViewVertex*. Com o *ViewMap* nós também podemos calcular imagens auxiliares tal como um *buffer* de profundidade.

O módulo de *Shader* é o processo de desenho de linha programável. Ele toma como entrada o *view map* e as imagens auxiliares, e cria os strokes que compõe o desenho. Os strokes são descritos por uma estrutura unidimensional e um conjunto de atributos que variam ao longo do comprimento dele (densidade, cor, transparência, textura, etc.). Nossa escolha de *ViewEdges* como o elemento atômico no *ViewMap* nos dá um compromisso excelente entre a generalidade (é tranquilamente possível a criação de strokes múltiplos para descrever um único ViewEdge) e solidez (quando possível ou necessário um stroke pode atravessar ViewEdges múltiplas, por exemplo um único stroke pode descrever o contorno externo de um objeto).

Finalmente, o sistema de marcas é responsável pelo renderização atual

de strokes primitivos com os atributos deles. Por exemplo, o mesmo stroke com dada densidade e cor pode ser renderizado com estilos de marcas diferentes tal como lápis de cor ou pintura à óleo.

O conjunto de processos que implementam um estilo pintado é chamado de estilo *sheet*. Ele é usualmente decomposto em uma série de módulos de estilo que são responsáveis por subpartes do desenho. A utilização de módulos de estilo é um meio usado para variar o estilo do interior do desenho. Por exemplo, o objeto principal pode ser desenhado usando um módulo de estilo diferente do resto da cena, ou linhas escondidas podem ser desenhadas com um módulo de estilo diferente das visíveis.

A operação de um módulo de estilo consiste principalmente em controlar as características topológicas dos strokes, assim como atributos visuais deles. Experimentações com desenhos estilizados conduz à definição do próximo conjunto de operadores topológicos, o qual foi incluído para ser suficiente na criação de todos os efeitos estilísticos. Primeiro, a seleção de um conjunto de ViewEdges, que é o mecanismo usado para limitar atenção para um subconjunto de *ViewEdges* de certo desenho. O operador de encadeamento permite construir uma seqüência unidimensional de ViewEges, iniciando por uma dada aresta na seleção. Depois, o operador de divisão permite refinar os elementos do desenho em cadeias quebradas em locais apropriados (por exemplo, pontos de curvatura alta). Finalmente, uma última classe de operadores determinam atributos para os strokes (por exemplo, cor, largura, textura, transparência, etc.). Estes operadores são inteiramente programáveis, e são aplicados interativamente em um modo de *pipeline*.

4.2 Módulo de Extração

No módulo de extração, nós extraímos as linhas características do modelo, tal como silhuetas e contornos sugestivos. Também foi calculada a visibilidade das arestas. Essa extração foi feita da seguinte forma:

Primeiro extraímos todas as linhas relevantes do modelo, com respeito à área de visão. Nós calculamos as silhuetas da seguinte forma: para cada aresta, calcula-se as normais das faces adjacentes a ela e se o ângulo entre o vetor de visão e a normal de cada uma das duas faces tem sinais trocados, então essa aresta é uma aresta silhueta (O segundo método de extração de silhuetas discutido na seção 2.2). A figura 4.2 mostra um exemplo de extração da silhueta em um modelo 3D da Deusa Athena.

Os contornos sugestivos foram calculados achando o ângulo entre as

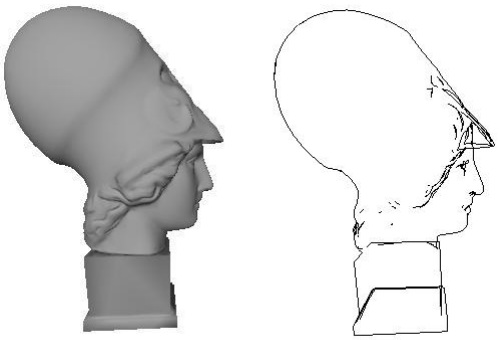

Figura 4.2: Silhueta extraída do modelo 3D da deusa Atenas.

normais das duas faces adjacentes a cada aresta da malha. Se esse ângulo for menor do que certo ângulo predeterminado, dizemos que essa aresta é um contorno sugestivo (o segundo método também apresentado na seção 2.2. A figura 4.3 nos mostra os contornos sugestivos extraídos do modelo 3D da Deusa Athena.

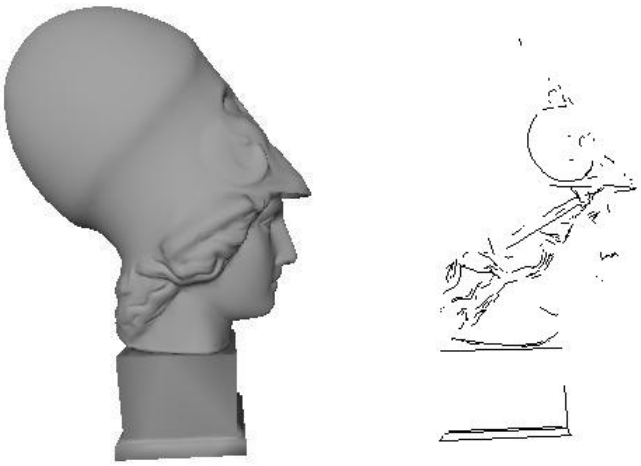

Figura 4.3: Contornos Sugestivos do modelo 3D.

O calculo das arestas escondidas foi feita usando a forma intuitiva apresentada na seção 2.2. Na figura 4.5 vemos um exemplo de um cubo que foi aplicado esse tratamento de arestas escondidas nos contornos sugestivos previamente extraídos: na esquerda, o cubo sem o tratamento e na direita, os mesmo contornos agora com o tratamento de arestas escondidas.

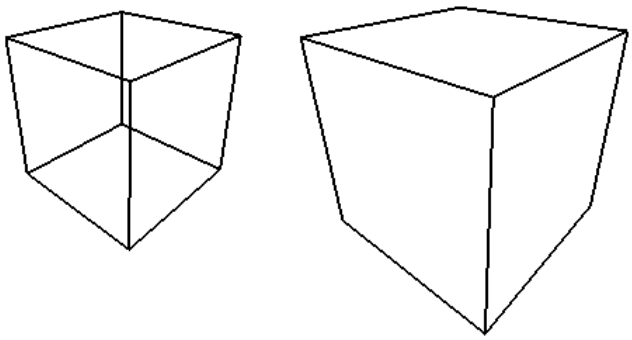

Figura 4.4: Tratamento de arestas escondidas dos contornos sugestivos de um cubo.

4.3 Módulo de Shader

Na nossa implementação, após ter extraído as silhuetas e contornos sugestivos e ter feito um tratamento de arestas escondidas, conseguimos transformá-las em strokes. Depois disso, conseguimos aplicar atributos de cor e espessura a esses strokes individualmente. Na figura 4.5 vemos um exemplo de uma garça onde foi extraído as linhas características e foi aplicado atributos de cor e espessura de linha.

Figura 4.5: Foi dada a silhueta a cor rosa e espessura 3. Já aos contornos sugestivos, foi aplicada uma cor azul e espessura 1.

No seu trabalho, Grabli apresentou também *módulos de estilos programáveis*, mas a nossa implementação não abrangeu tais módulos. Esses módulos são divididos em dois: operadores de criação de strokes e operadores de atributos. Eles permitem a especificação de regras que governam o processo de desenho para topologia e atributos dos strokes. Tais módulos são aplicados seqüencialmente para obter sucessivas camadas no desenho. Cada módulo de estilo trabalha no viewmap e produz um conjunto de strokes através de uma organização de pipeline. Alguns dos operadores

(por exemplo, Seleção) aplica-se a qualquer elemento 1D e pode ser usado em qualquer estágio do pipeline, enquanto que outros (por exemplo, Encadeamento) somente aplica-se a um tipo específico de elementos e deve então ser chamado em um local específico no pipeline. Vamos primeiro discutir os operadores que tratam da criação topológica dos strokes. Então, discutiremos a última classe de operadores, a designação de atributos. Um módulo de estilo é construído como uma seqüência de chamadas para esses operadores. Na figura 4.6 (ver [1]) podemos ver cada passo do pipeline desses módulos de criação de strokes.

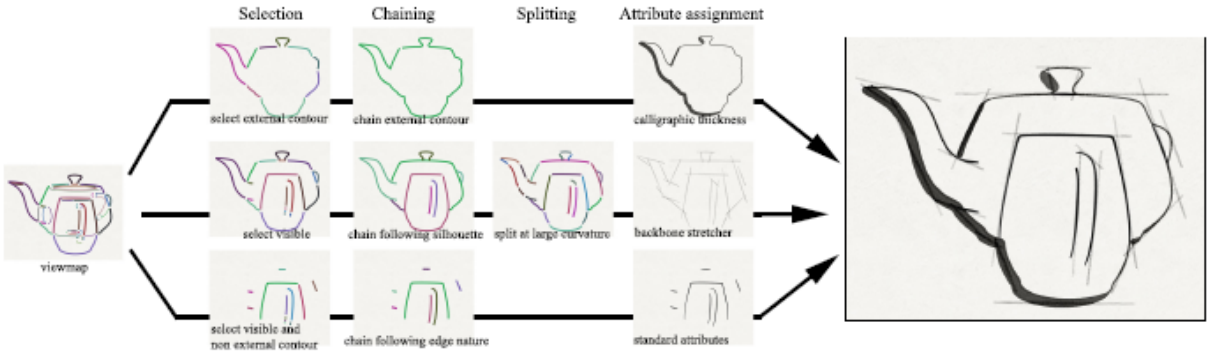

Figura 4.6: Exemplo dos operadores propostos por Grabli onde são usados dentro dos três módulos de estilo constituindo um estilo sheet simples.

Vamos discutir cada um desses módulos:

Seleção A seleção é fundamental para dividir o desenho em camadas, para aplicar um estilo a uma subparte das linhas características ou para omitir linhas que são desnecessárias. É fornecido um mecanismo de seleção através de um operador de seleção. Este operador trabalha em elementos 1D (*ViewEdges*, Cadeias ou strokes). Na prática, um operador de seleção extrai um subconjunto de conjuntos ativos de elementos 1D para definir um novo conjunto ativo. A regra de seleção é especificada como um predicativo de um elemento 1D. Por exemplo, a seleção pode ser baseada em invisibilidade quantitativa, na ID do objeto ou na natureza (dobras, silhuetas ou bordas). Esses predicados podem ser combinados com operadores lógicos clássicos. Pode ser também implementados novos predicados baseados em funções mais complexas da cena. O operador de seleção é principalmente usado no começo do *pipeline* de programação. No entanto, como mencionado antes, ele pode ser utilizado em algum outro estágio do *pipeline*, e pode ser útil para refinar a seleção depois do encadeamento, quando informações sobre a topologia potencial de um stroke é acessível.

Encadeamento A área de visão de um objeto, como codificado no *viewmap*, fornece informações do grafo, enquanto que desenhos de linhas consistem de linhas 1D. A escolha de uma linha e comprimento do stroke tem repercussão na aparência do desenho. Note que aqui pode ser usado vários strokes para representar a mesma linha característica, e que strokes podem transpor várias linhas características (veja o contorno externo na figura 4.6). Quando criado um stroke, tem-se que identificar duas espécies de decisões: primeiro, deve-se decidir para cada vértice do grafo qual caminho seguir. Segundo, deve-se decidir onde iniciar ou parar um strokes. Nesta aproximação, a forma é tratada por operadores de encadeamento, e o segundo por operadores de divisão.

Os operadores de encadeamento criam listas conectadas de *ViewEdges*, no qual se chama cadeias. Um operador de encadeamento é executado sucessivamente em todas as *ViewEdges* na seleção, e constroem uma cadeia originando de cada uma, opcionalmente pode ligar cada *ViewEdges* do jeito que ela é processada. Uma regra de encadeamento deve responder duas questões: quando parar e onde voltar a um *View Vertex*.Para essa regra escolhe-se um iterador. O método de incrementação deste iterador decide qual é a próxima *ViewEdge* entre as adjacentes ao *ViewVertex*. O critério de parada do iterador decide se a cadeia deve ser parada. Por exemplo, ela pode parar quando certo comprimento é alcançado, se uma oclusão é encontrada ou quando a curvatura é alta demais.

Estilo de desenho pode permitir que vários strokes sobreponha-se ou não. A sobreposição de strokes pode ser útil para processar o aspecto esboçado como ilustrado na figura 4.7 (c) obtida por Grabli. É fornecido um mecanismo de etiquetagem para controlar ou prevenir múltiplo encadeamentos da mesma *ViewEdge*. Além disso, o encadeamento pode ser bidirecional ou unidirecional. O encadeamento pode também ser forçado a ficar dentro da seleção ou não. No segundo caso, cada cadeia inicia em um *ViewEdge* da seleção, mas elas pode conter *ViewEdges* arbitrárias. Isto pode ser útil, por exemplo, para selecionar somente arestas na silhueta externa de um objeto, mas permite que o encadeamento estenda-se a uma silhueta interna (não selecionada) como mostrada na figura 4.7. O sistema deve fornecer várias estratégias de encadeamento padrão, tal como: encadeamento de *ViewEdges* de mesma natureza, contorno próximo ou contornos externos. Na figura 4.7 podemos ver um exemplo de en-

cadeamento simples, aplicados a um conjunto de ViewEdges do contorno externo de um desenho:**(a)** segue o contorno externo, **(b)** segue silhuetas no mesmo objeto, **(c)**segue silhuetas no mesmo objeto e permite múltiplos encadeamentos da mesma *ViewEdges*. Note como no caso **(b)** e **(c)** o operador de encadeamento engloba arestas que não fizeram parte da seleção original.

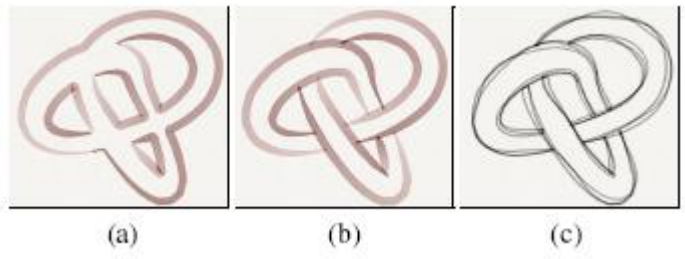

Figura 4.7: Exemplo de predicativos de encadeamento simples, aplicados a um conjunto de *ViewEdges* no contorno externo de um desenho. Resultados obtidos por Grabli.

Divisão de cadeias Como discutido anteriormente, uma cadeia pode ser descrito com vários strokes de tamanho pequeno. Isto é a função do operador de divisão de cadeias. Ele recebe uma cadeia como entrada e cria um número de strokes que as descreve. A regra dada para este operador é principalmente decidir onde a cadeia deve ser dividida.

Podem ser desenvolvidas duas estratégias diferentes para selecionar o comprimento de strokes e pontos de quebra: a divisão seqüencial e a recursiva. Na divisão seqüencial, atravessa-se a cadeia seqüencialmente e a decisão de dividir é baseado em um predicativo tal como comprimento máximo, natureza do vértice ou densidade local. Este mecanismo é fácil de especificar mas baseia-se somente na informação local. Por outro lado, a divisão recursiva toma uma decisão global na cadeia e divide recursivamente ao longo do mínimo da função especificada pelo usuário.

É importante notar que talvez se precise dividir a cadeia em outro lugar que *ViewVertices* ou vértices do modelo de entrada. Vértices temporários neste valor de amostra são criadas interativamente a medida que a cadeia é atravessada, mas eles não são guardados permanentemente.

Ordenação Nesta aproximação, a seqüência na qual *ViewEdges*, Cadeias e strokes são tratados podem influenciar o desenho: no operador de encadeamento por exemplo, um mecanismo de **timestamp** previne o

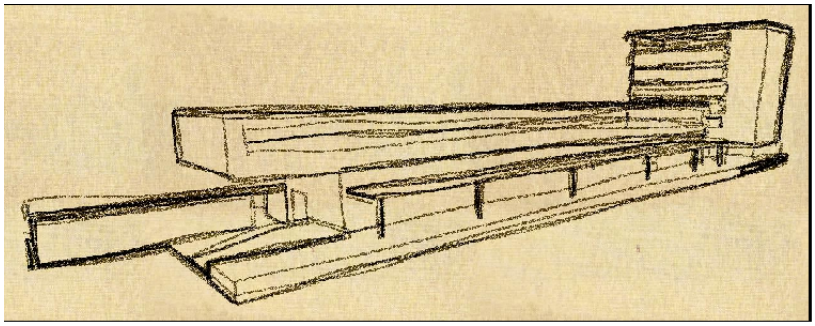

Figura 4.8: Uso de múltiplos strokes através de cadeias para renderizar construção em um estilo esboçado. Resultado também obtido por Grabli

reuso de *ViewEdges*. Além disso, a informação de densidade do stroke evolui-se com o desenho atual. Por exemplo, quando a informação de densidade é usada para prevenir confusão, ele é importante para processar *ViewEdges* que são visualmente mais importante primeiro, de forma que elas são menos provável a serem omitidas. Dessa forma, fornece-se um operador de ordenação, o qual permite a classificação de um elemento 1D qualquer. A regra de ordenação é expressa como um predicativo de comparação baseado em comprimento, importância, profundidade, variação da profundidade local, localização espacial 2D, etc. A definição de uma ordenação relevante de *ViewEdges* ou strokes pode ser muito entediante e requer a avaliação e integração de diferentes espécies de informações que podem somente ser especificado usando uma aproximação programável.

Atribuição de atributos Depois de criar os strokes, precisamos determinar seus atributos tal como cor, densidade, transparência e localização espacial. Este passo é o mais semelhante ao sistema de *shading* tradicional, exceto que nós operamos em strokes unidimensional, particularmente em fragmentos 0-dimensional. Note que os strokes de um desenho freqüentemente não seguem exatamente a geometria auxiliar. Isto é porque o *shaders* proposto por Grabli pode modificar a localização espacial dos pontos principais. Isto tem um sentido semelhante à técnica de mapeamento de mudança de local.

Os nomeadores de atributos são implementados como processo trabalhando em strokes. Como discutido anteriormente, algumas informações podem ser questionada no stroke para propor decisões estilísticas dentro do processo. Como um caso espacial, é utilizado operadores de atributos para deletar strokes em ordem para evitar desordem.

Operados de designação de atributos múltiplos pode ser aplicados para um stroke seqüencialmente. Isto é útil para controlar atributos diferentes. Um operador determina cor enquanto que um segundo determina a densidade. Além disso, atributos podem ser determinados em uma maneira absoluta (o valor previsto é trocado) ou em uma maneira relativa (o valor previsto é adaptado).

Os atributos previamente determinados são interpolados na nova localização. Um número de operados atômicos em strokes são avaliados e podem ser usados no contexto de modificação da geometria do strokes.

Operadores simples tal como atribuição de atributos constantes são fornecidos. Um operador de atributo especial determina o estilo da marca usada para a renderização do stroke. Outro shader simples inclui um "removedor de ponta" que adaptam a parte inicial e final do stroke. Isto permite, por exemplo, o clássico "*line haloing*" para melhorar percepção de profundidade. Além disto, várias técnicas padrões úteis que tem sido usado há muito tempo em RNF para efeitos *sketchy*, tal como ruídos, deslocamento de strokes ou suavização, por exemplo [3, 18] são fornecidos como componente básico para o sistema.

Densidade e Omissão Omissão de strokes é uma ferramenta crucial para prevenir desordem visual ou produzir estilos minimalistas. Um jeito para alcançar isso é criar condicionalmente strokes somente quando a densidade local na região afetada do desenho é suficientemente baixa. Assim, são criados menos strokes em partes comprimidas da imagem. A própria omissão de strokes conta com duas componentes maiores: uma deve estimar a densidade visual em regiões do desenho e um deve dar prioridade a strokes para que certo stroke "importante" seja desenhado primeiro para evitar omissão dele devido à densidade excessiva. Winkenbach, Salisbury e outros resolveram o segundo problema no caso espacial de áreas de textura [12], pode-se então inserir algum operador de ordem para controlar o processo.

O sistema proposto por Grabli permite questionar a densidade do desenho atual em um dado ponto e escala. Esta função pode ser usada, por exemplo, dentro de um predicativo threshold simples e descartar candidatos a strokes quando o desenho é desordenado nesta vizinhança. O mecanismo descrito é causal. No sentido em que a densidade é avaliada baseada no que tem sido desenhado distante. Em muitos casos é necessário ter uma idéia da densidade potencial do desenho,

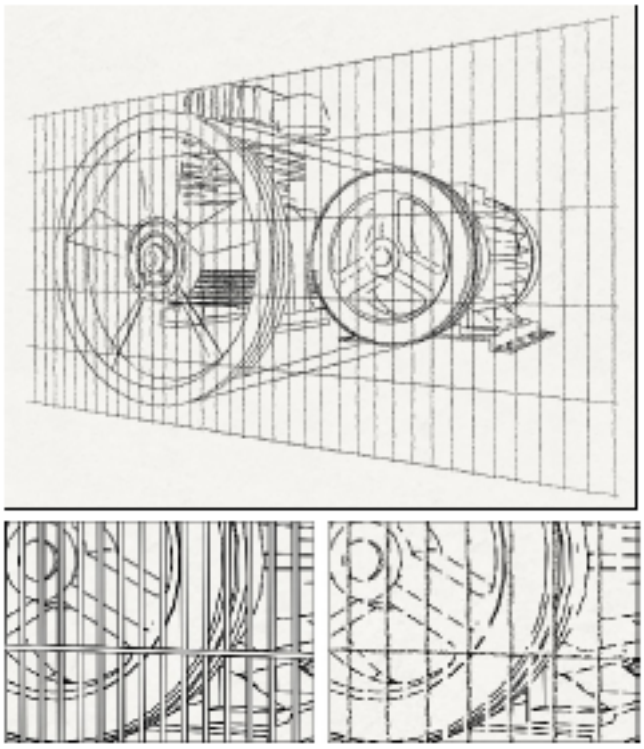

Figura 4.9: Topo: Desenho simplificado de uma área de visão complexa. Inferior esquerdo: visão close-up com todas as linhas visíveis mostrando a complexidade visual não desejada. Resultado obtido por Grabli.

se todas as linhas são desenhadas. Isto é obtido do mapa de densidade de visão pré-calculada. Note que a combinação de estimativas de densidades em escalas diferentes fornece muitas informações ricas sobre a complexidade local. Informação de densidade pode ser usada também para modular atributos de strokes como densidade e cor.

A figura 4.9 ilustra o uso de um operador de encadeamento complexo bem como densidade causal para construir um representação simplificada de uma estrutura densa com oclusão. Para a rede uma cadeia é criada para cada barreira conectando todas viewedges incluindo uma oclusão curta. Essas cadeias são ordenadas por comprimento e subjetivado para o operador de densidade casual com um núcleo de variedade gaussiana e a classificação depende da profundidade. Assim permite-nos guardar somente um stroke simples para cada barreira e remover exatamente no meio das barreiras. O compressor atrás da rede também usa um iterador de encadeamento avançado para evitar o efeito de linha tracejada mostrada na imagem de fundo direita.

Marcas da parte de trás O sistema de marca é ortogonal à aproximação de desenhos de linha programáveis proposto por Grabli. O desenvolvi-

mento de uma marca da parte de trás programável é algo muito importante. O sistema apresentado de renderização de marcas usa a API OpenGL padrão. Strokes são renderizados como faixa de triângulos, determinados pelo objeto principal e amostras de densidade. Técnicas padrões são usadas para prevenir singularidades da curva compensada em curvatura alta.

É usado texturas de strokes reais como mapas alfa para aumentar a qualidade visual, o uso de transparência sozinha permite-nos controlar a cor de cada traço, como especificado por seus atributos. Nós usamos OpenGL combinando modelos para imitar vários tipos físicos medianos. Na prática, nós renderizamos o inverso da imagem, assim que uma tela branca corresponde (0,0,0). Isto facilita o uso de combinações e a simulação da natureza subtrativa da mídia maior. Nós usamos uma forma de troca simples para mídia grosseira tal como pintura a óleo. A combinação aditiva (o qual se torna subtrativo em nosso contexto inverso) é bem-adaptado para materiais úmidos tal como tinta. Finalmente, o modo de combinação mínima provido por OpenGl 1.2 podem imitar grafite e outras mídias secas. Uma textura de fundo pode se aplicado. Ela é, porém renderizada somente para o desenho final e não afeta o cálculo de densidade.

Capítulo 5

Resultados

Nós utilizamos a linguagem C++ para implementar alguns resultados. O nosso programa possui um interface que permite ao usuário extrair as linhas características de um arquivo com extensão *ply* e atribuir a essas linhas novas características como cor, espessura da linha e opacidade.

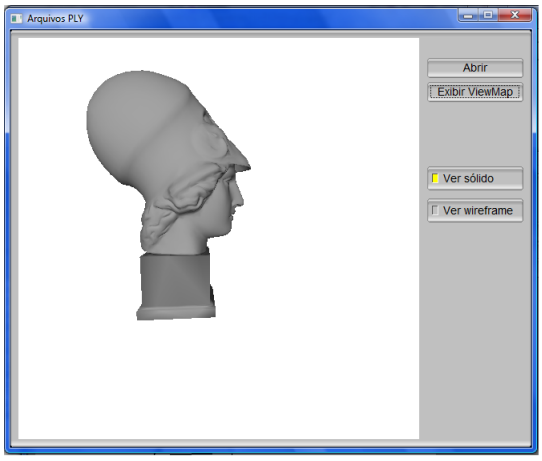

Figura 5.1: Janela de visualização dos arquivos *ply*.

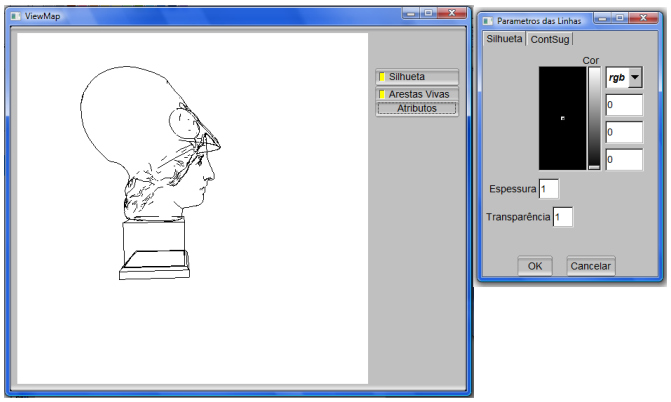

Figura 5.2: (Esquerda)Janela de visualização do ViewMap. (Direita) Janela onde o usuário pode modificar os atributos da silhueta e dos contornos sugestivos separadamente.

A figura 5.3 mostra a extração do ViewMap de um objeto simples como cubo e a aplicação de atributos de cor e espessura de linha diferentes para a silhueta e arestas vivas.

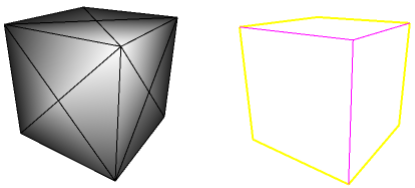

Figura 5.3: Aplicamos a cor amarela e espessura 2 para a silhueta e cor rosa e espessura 1.5 para os contornos sugestivos.

Na figura 5.4 exibe um exemplo que dá ênfase a silhueta do modelo 3D da deusa Athena.

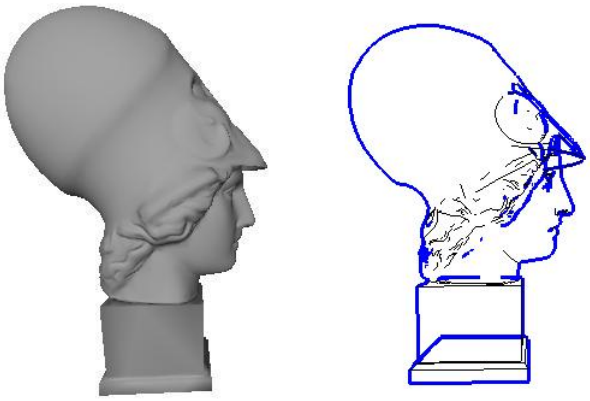

Figura 5.4: Neste exemplo damos ênfase a silhueta do objeto mudando a sua cor para azul e aumentando a espessura da linha para 2.

Na figura 5.5 destacamos os contornos sugestivos para mostrar a sua importância no desenho 2D do modelo 3D de um tênis.

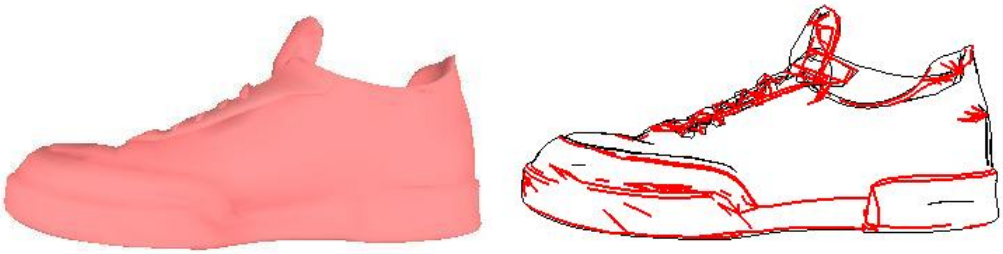

Figura 5.5: Foi dada ao contorno sugestivo do tênis uma cor vermelha e espessura 2.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Este trabalho discutiu os principais conceitos sobre a renderização não-fotorrealística, suas aplicações e os principais métodos. Discutimos principalmente métodos de RNF aplicados a imagens e a modelos 3D. O trabalho apresentou um esquema geral que pode ser usado para o desenvolvimento e implementação de métodos de renderização não-fotorrealística, que englobando tanto a RNF de imagens quanto a RNF de modelos 3D. Finalmente, este trabalho discutiu a implementação de um método de RNF para criar estilos diferentes de contornos de objetos representados por malhas triangulares.

Dentre os métodos estudados, foi possível fazer uma classificação dos tipos de renderização como renderização de imagem ou renderização de objetos. Dentro da renderização de imagens ainda podemos classificar métodos que tentar inferir uma geometria 3D no objeto e métodos que simplesmente buscam mudar a aparência visual de uma imagem. Nós concentramos em métodos onde temos como entrada um *cartoon* feito a mão e se extrai dele curvas características e normais para possíveis aplicações de efeitos de contorno e preenchimento. Já nos métodos para modelos 3D, utiliza-se, na maioria das vezes malhas triangulares, onde são extraídas curvas características para serem aplicados, assim como nas imagens, efeitos de contornos e de área. Esse estudo nos levou a apresentar um esquema geral para a RNF.

Neste esquema apresentamos os módulos que compõe o desenvolvimento de uma RNF. O primeiro módulo apresentado é o módulo objeto que toma como entradas os modelos 3D ou imagens. Em seguida o módulo extração que visa extrair informações sobre as curvas e normais do modelo 3D ou imagem. Essa casse é uma classe abstrata pois depende do objeto de en-

trada para calcular essas curvas, já que são calculadas de maneira diferente para modelos e imagens. Depois de extraídas essas informações, passamos para o módulo de shader onde definimos dois submódulos: módulo de contornos e módulo de áreas. O módulo de contorno tem o objetivo de aplicar RNF nos contornos do objeto, tal como espessura e cor. Já o módulo de áreas aplica efeitos nas áreas do objeto tal como preenchimento. Depois de todos esses passos obtemos uma imagem renderizada.

Também foi discutido o trabalho de Gabri [1] sobre um método de renderização não-fotorealística programável em desenhos de linhas. Ele também apresentou um esquema de renderização, só que voltado para a renderização de desenhos de linhas, que se enquadra no esquema que nós apresentamos. Nós implementamos alguns passos do método proposto por ele, obtendo um programa onde foi possível fazer extração de silhuetas e contornos sugestivos de malhas triangulares 3D e aplicá-los atributos de cor e espessura.

6.2 Trabalhos Futuros

- Estender o módulo de estilos para permitir a criação de novos estilos através de edição de outros atributos.
- Implementar a arquitetura proposta no esquema geral.
- Implementar os operadores de criação de strokes propostos por Grabli: seleção, encadeamento, divisão de cadeias e ordenação.
- Estender o esquema geral para suportar animação de *cartoon*, acrescentando módulos de câmera, módulos de transformações para módulos de interpolação (keyframes).

Referências Bibliográficas

[1] GRABLI, S.; TURQUIN, E.; DURAND, F. ; SILION, F.. Programmable style for npr line drawing. In: PROCEEDINGS OF NPAR - 3RD INTERNATIONAL SYMPOSIUM ON NON-PHOTOREALISTIC ANIMATION AND RENDERING, p. 000, 2004.

[2] FUN, I.L.S.; SUNAR, M.S.; BADE, A. *Non-Photorealistic Outdoor Scene Rendering: Techniques and Application*. Intenational Conference on Conputer Graphics, Imaging and Visualization (2004).

[3] KALNINS,R.; MARKOSIAN, L.; MEIER, B.; KOWALSKI, M.; LEE, J.; DAVIDSON, P.; WEBB, M.; HUGHES,J. FINKELSTEIN, A. *WYSIWYG NPR: Drawing Strokes Directly on 3D Models*. SigGraph (2002).

[4] DECARLO, D., FINKELSTEIN, A.; RUSINKIEWICZ, S.;SANTELLA, A. *Suggestive contours for conveying shape*. ACM Trans. on Graphics 22, 3 (2003).

[5] HERTZMANN, A. *Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines*. Proc. SIGGRAPH (1999).

[6] APPEL, A. *The notion of quantitative invisibility and the machine rendering of solids*. Proc. ACM Natl. Mtg. (1967), 387. 5

[7] WILLATS, J. *Art and Representation*. Princeton U. Pr.,1997. 3, 6.

[8] ISENBERG, T., FREUDENBERG, B., HALPER, N.,SCHLECHTWEG, S., STROTHOTTE, T. *A developers guide to silhouette algorithms for polygonal models*. IEEE Computer Graphics and Applications special issue on NPR (2003).

[9] GOOCH, B.; SLOAN P.; GOOCH, A.; SHIRLEY, P.;RIESENFELD, R. *Interactive Technical Illustration*.ACM Symp. on Interactive 3D Graphics (1999).

[10] BEZERRA, H.; FEIJÓ, B.; VELHO, L.: *Colorizaçã 3D para Animação 2D*. Rio de Janeiro, 2005. 75p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[11] HAMEL, J.; STROTHOTTE, T. *Capturing and re-using rendition styles for non-photorealistic rendering*. In Proc. Eurographics (1999).

[12] WINKENBACH G., SALESIN D.: *Computer-generated pen-and-ink illustration*. Proc. SIGGRAPH (1994).

[13] PLOTZE, R. O.; BRUNO, O. M. *Estudo e comparação de algoritmos de esqueletonização para imagens binárias*. IV Congresso Brasileiro de Computação - CBComp 2004.

[14] DECARLO, D.; SANTELLA, A. *Stylization and Abstraction of Photographs*. In SIGGRAPH (2002).

[15] HSU, S. C., LEE, I. H. H. *Drawing and animation using skeletal strokes*. Proc. SIGGRAPH 94 (1994).

[16] STROTHOTTE, T., SCHLECHTWEG, S. *Non-Photorealistic Computer Graphics. Modeling, Rendering, and Animation*. Morgan Kaufmann, 2002.

[17] MARKOSIAN, L., KOWALSKI, M., TRYCHIN, S., BOURDEV, L., GOLDSTEIN, D., HUGHES, J. *Realtime nonphotorealistic rendering*. Proc. SIGGRAPH (1997).

[18] SOUSA, M., PRUSINKIEWICZ, P. *A few good lines: Suggestive drawing of 3d models*. Computer Graphics Forum (Proc. of EuroGraphics '03) 22, 3 (2003).

[19] WOO, M., ET AL.: OpenGL programming guide: the oficial guide to learning OpenGL, ver. 1.2, 3ª ed. Addison-Wesley, Reading, MA, USA, 1999.

[20] PARKER, J. R. *Pratical Computer Vision using C*. John Wiley and Sons, Inc, New York, 1994.

[21] http://student.kuleuven.be/ m0216922/CG/floodfill.html#Scanline_Floodfill_Algorithm_With_Stack.

[22] DURAND, F., OSTROMOUKHOV, V., MILLER, M., DURANLEAU, F., DORSEY, J. 2001. *Decoupling strokes and high-level attributes for interactive traditional drawing.* In 12th Eurographics Workshop on Rendering, 71-82.

[23] CORRÊA, W. T.; JENSEN, R. J.; THAYER, C. E. ; FINKELSTEIN, A.. Texture mapping for cel animation. In: PROCEEDINGS OF SIGGRAPH, p. 435-446, 1998.

[24] PEIXOTO, A.; CARVALHO, P.C.; *Esqueletos de Objetos Volumétricos.* Monografias em Ciência da Computação, PUC-RIO, 2000.

[25] OGNIEWICZ, R.L. *Discrete Voronoi Skeletons.* Hartung-Gorre, 1993.

[26] ZHOU, Y.; TOGA, A.W. *Efficient Skeletonization of Volumétric Objects.* IEEE Transactions on Visualization and Computer Graphics, 5,3,196-209, 1999.

[27] www.inf.ufsc.br/ patrec/imagens.html

[28] JOHNSTON,S. F: *Lumo: illumination for cel animation.* InNPAR 2002: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering, pages 45-ff, New York, USA, 2002. ACM Press.