Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação

Renderizações Não Fotorealísticas para Estilização de Imagens e Vídeos Usando Areia Colorida

Laurindo de Sousa Britto Neto

Natal/RN
Setembro de 2007

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação

Renderizações Não Fotorealísticas para Estilização de Imagens e Vídeos Usando Areia Colorida

Exame de dissertação submetido ao Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como parte dos requisitos para a obtenção do grau de Mestre em Sistemas e Computação (M.Sc.).

Laurindo de Sousa Britto Neto

Natal/RN
Setembro de 2007

Divisão de Serviços Técnicos

Catalogação da Publicação na Fonte. UFRN / Biblioteca Central Zila Mamede

Britto Neto, Laurindo de Sousa.

Renderizações não fotorealísticas para estilização de imagens e vídeos usando areia colorida / Laurindo de Sousa Britto Neto. – Natal, RN, 2007.

74 f.

Orientador : Bruno Motta de Carvalho.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Centro de Ciências Exatas e da Terra. Departamento de Informática e Matemática Aplicada. Programa de Pós-Graduação em Sistemas e Computação.

1. Computação gráfica – Dissertação. 2. Vídeos de areia – Dissertação. 3. Texturas procedurais – Dissertação. 4. Coerência temporal – Dissertação. I. Carvalho, Bruno Motta de. II. Título.

RN/UF/BCZM CDU 004.92 (043.3)

Renderizações Não Fotorealísticas para Estilização de Imagens e Vídeos Usando Areia Colorida

Laurindo de Sousa Britto Neto

Este exame de dissertação foi avaliado e considerado aprovado pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Bruno Motta de Carvalho
Orientador

Profa. Dra. Thaís Vasconcelos Batista
Coordenador do Programa

Banca Examinadora:

Prof. Dr. Bruno Motta de Carvalho
Presidente

Prof. Dr. Luiz Marcos Garcia Gonçalves

Prof. Dr. Marcelo Walter

"Uma imagem vale mais do que mil palavras."
provérbio chinês

Agradecimentos

Agradeço primeiramente a meus pais, Laurindo de Sousa Britto Júnior e Carmen do Monte de Carvalho Britto, pelo apoio, incentivo e confiança que tiveram durante todo o período do mestrado e durante toda a minha vida. A meu irmão, Felipe de Carvalho Britto, por me incentivar e querer sempre que eu cresça cada vez mais. A família é à base da vida. Sem ela seria mais difícil conseguir ultrapassar as barreiras e etapas que a vida nos impõe. Agradeço também a meus tios, primos e amigos de Teresina/PI, por me ligarem sempre nos fins de semana, em especial ao meu primo André Carvalho Luz.

Agradeço ao Prof. Dr. Bruno Motta de Carvalho pela oportunidade, orientação, paciência e amizade despendida durante o período do mestrado. Um grande professor que admiro como profissional e me sinto privilegiado por ter sido seu orientando. Aos professores Dr. Luiz Marcos Garcia Gonçalves (DCA/UFRN), Dr. Marcelo Walter (CIn/UFPE) e Dr. Selan Rodrigues dos Santos (DIMAp/UFRN) pela avaliação da dissertação através de críticas, correções e sugestões, contribuindo com a qualidade do trabalho.

Um agradecimento especial aos amigos feitos durante o período do mestrado: Lucas Oliveira, Raul Paradeda, Fred Lopes, Milano Gadelha, Diogo Fagundes, Íria Caline, Rodrigo Gaete, Natal Henrique, Eduardo Marcondes, Gilbran Andrade, Cláudia Fernanda, Araken Medeiros, Shirlly Christiany, Valéria Siqueira, Márjory Abreu, Carlos Eduardo, Lígia Bagi e Cristine Schmidt.

Agradeço aos membros da equipe AnimVideo, aos companheiros de Lablic, ao pessoal do PPgSC e aos professores do DIMAp. Agradeço a Rafael Gomes por ceder uma animação de sua autoria para realização de experimentos. E por fim, agradeço ao CNPq pelo apoio financeiro durante boa parte do mestrado.

Resumo

Renderização Não Fotorealística (NPR) é uma classe de técnicas que almejam reproduzir técnicas artísticas, tentado expressar sentimentos e emoções nas cenas renderizadas, dando um aspecto de que foram feitas "manualmente". Outra forma de definir a NPR é como o processamento de cenas, imagens ou vídeos para geração de trabalhos de arte, gerando cenas, imagens ou vídeos que podem ter o atrativo visual de peças artísticas, expressando características visuais e emocionais do estilo artístico. Esta dissertação apresenta um novo método de NPR para estilização de imagens e vídeos baseado em uma expressão artística típica da região Nordeste do Brasil, que usa areia colorida para compor imagens de paisagens na superfície interna de garrafas de vidro. Este método possui uma técnica para geração de texturas procedurais de areia 2D, e duas técnicas que imitam os efeitos criados pelos artesões usando sua ferramenta. Além disso, essa dissertação apresenta também um método para geração de animações $2\frac{1}{2}$D em caixas de areia a partir do vídeo estilizado. A coerência temporal nos vídeos estilizados pode ser forçada nos objetos individuais do vídeo com auxílio de um algoritmo de segmentação de vídeo. As técnicas apresentadas neste trabalho são usadas na estilização de vídeos reais e sintéticos, algo quase impossível de ser produzido pelo artesão na vida real.

Área de Concentração: Teoria e Inteligência Computacional.
Palavras-chave: computação gráfica, renderizações não fotorealísticas, texturas procedurais, renderização estilizada, estilização de vídeo, coerência temporal, vídeos de areia.

Abstract

Non-Photorealisitc Rendering (NPR) is a class of techniques that aims to reproduce artistic techniques, trying to express feelings and moods on the rendered scenes, giving an aspect of that they had been made "by hand". Another way of defining NPR is that it is the processing of scenes, images or videos into artwork, generating scenes, images or videos that can have the visual appeal of pieces of art, expressing the visual and emotional characteristics of artistic styles. This dissertation presents a new method of NPR for stylization of images and videos, based on a typical artistic expression of the Northeast region of Brazil, that uses colored sand to compose landscape images on the inner surface of glass bottles. This method is comprised by one technique for generating 2D procedural textures of sand, and two techniques that mimic effects created by the artists using their tools. It also presents a method for generating $2\frac{1}{2}$D animations in sandbox from the stylized video. The temporal coherence within these stylized videos can be enforced on individual objects with the aid of a video segmentation algorithm. The present techniques in this work were used on stylization of synthetic and real videos, something close to impossible to be produced by artist in real life.

Area of Concentration: Theory and Computational Intelligence.
Key words: computer graphics, non-photorealistic rendering, procedural textures, stylized rendering, video stylization, temporal coherency, sand movies.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Renderização Fotorealística (*Photorealistic Rendering*) é o processo de criação de uma imagem sintética, a partir de informações de ambientes tridimensionais (3D), como sua geometria e o tipo de material que compõem os objetos pertencentes a cena [3], para que através de um conjunto de técnicas algorítmicas e as leis da física envolvidas no processo fotográfico [71], a imagem se aproxime ao máximo possível de uma fotografia real.

A *Renderização Não Fotorealística* (*Non-Photorealistic Rendering* ou NPR), como o próprio nome diz, é uma classe de técnicas definida negativamente a partir da renderização fotorealística. Seu objetivo é a criação de técnicas alternativas à renderização de imagens que buscam reproduzir cenas fotorealísticas. Outra forma de definir a NPR é o processamento de cenas, imagens ou vídeos em trabalhos de arte, gerando cenas, imagens ou vídeos que podem ter o atrativo visual de peças de arte, expressando características visuais e emocionais de estilos artísticos [12]. Recentemente, vários autores tem adotado o termo renderização estilizada para definir esta classe de técnicas de uma forma mais expressiva.

Juntando ciência e arte, a NPR dá uma ênfase maior na comunicação do conteúdo de uma imagem, através do efeito artístico que a imagem final deve possuir. Deste modo, técnicas que são utilizadas há bastante tempo por artistas podem ser adaptadas à Computação Gráfica, eliminando-se assim, as informações indesejadas e/ou enfatizando determinadas características da imagem original.

Nesta dissertação desenvolvemos um método de renderização não fotorealística para a estilização de imagens e vídeos com areia colorida denominado *Csand*, do inglês "*colored sand*". Esse método é baseado na confecção artesanal de garrafas com areia colorida, um estilo artístico típico da região Nordeste do Brasil, desenvolvido principalmente nos estados do Rio Grande do Norte e Ceará. Desenvolvemos também uma técnica para a criação de animações $2\frac{1}{2}$D a partir do vídeo estilizado. (O termo $2\frac{1}{2}$D é usado devido à adição de informações 3D ao vídeo estilizado 2D, da mesma forma que o termo $2\frac{1}{2}$D foi utilizado por Snavely et al. [65] ao gerar vídeos estilizados que combinam informações 2D com 3D).

1.1 Motivação

Desde a década de 90, a NPR vem sendo uma área bastante ativa na Computação Gráfica. Nesta época, as pesquisas em NPR começaram a despertar um maior interesse na comunidade científica, onde conferências como o Eurographics '99 [11] e SIGGRAPH '98 [18] criaram sessões dedicadas ao tema. Ainda na década de 90, alguns livros clássicos de computação gráfica, que discutem antes de tudo sobre renderização fotorealística, surgiram com sessões específicas sobre aspectos da NPR, como por exemplo, Watt e Policarpo (1998) [78]. Em 2000, surgiu o NPAR '00 [25], o primeiro simpósio voltado exclusivamente para renderizações e animações não fotorealísticas. Posteriormente, foram publicados livros textos completos que tratam somente de assuntos relacionados à NPR como os livros de Gooch e Gooch (2001) [31] e de Strothotte e Schlechtweg (2002) [71].

Uma das vantagens da NPR é a possibilidade da utilização de especificações de modelos e métodos de animação mais simples do que na renderização fotorealística. A diminuição da complexidade da utilização de ferramentas de geração de animações pode ser associada a um aumento da utilização e impacto de tais ferramentas, possibilitando que usuários com um menor conhecimento de técnicas de animação e processos físicos que influenciam a criação de imagens fotorealísticas (como iluminação, reflexos e atenuação atmosférica) possam gerar animações, assim como usuários sem conhecimentos de técnicas artísticas possam fazer arte. Outras vantagens da NPR, em relação ao fotorealismo, são a diminuição do tempo de renderização devido à redução da complexidade do processo de renderização e a diminuição do espaço físico ocupado pelas imagens renderizadas devido a perda de informações da imagem causada pela NPR.

A NPR vem sendo usada em várias áreas como educação, arte e entretenimento. Grandes produções cinematográficas como "What Dreams May Come"(1998) [77] da Polygram Filmed Entertainment, fizeram uso de técnicas de renderizações não fotorealísticas. Ilustrações médicas e científicas, ilustrações técnicas, desenhos animados e outras aplicações usam técnicas de NPR para geração de suas imagens. Entretanto, as técnicas utilizadas atualmente ainda exigem uma interação muito grande com o animador, fazendo com que o mesmo tenha que editar um ou poucos quadros por vez. (Um outro exemplo de produção cinematográfica produzida usando estas técnicas é o filme "Waking Life"(2001) [47], comercializado pela Twentieth Century Fox Home Video).

A produção de garrafas com areia colorida é um estilo artístico típico da região Nordeste do Brasil, desenvolvido principalmente nos estados do Rio Grande do Norte e Ceará. A motivação desse trabalho é a criação desse estilo de expressão artística como um método de renderização não fotorealística para estilização de imagens e vídeos, além da divulgação dessa arte típica do Brasil em outros países através da publicação de artigos científicos.

O trabalho consiste na estilização de imagens e vídeos reais ou sintéticos, usando

areia virtual através da criação de texturas procedurais de areia colorida e na simulação de técnicas manuais usadas por artesãos com sua ferramenta na composição das imagens de paisagens formadas na superfície interna das garrafas de vidro. Essas texturas podem ser mapeadas dentro de modelos 3D como garrafas de vidro ou caixas de madeira. Além disso, foram geradas animações $2\frac{1}{2}$D a partir do vídeo estilizado, onde os objetos que se movem no vídeo deixam rastros e amontoados de areia por onde passam, na superfície plana de uma caixa de areia, movimentando os grãos de areia de áreas estáticas do vídeo.

Com a estilização do vídeo com areia colorida é possível criar garrafas com areia colorida fotorealisticamente parecidas a uma garrafa com areia colorida real, criar garrafas com imagens ou vídeos que seriam praticamente impossíveis de serem confeccionados manualmente pelos artesãos devido à complexidade de se gerar o número necessário de imagens coerentes entre si para criar um vídeo.

Esse trabalho faz parte do projeto *AnimVideo*, financiado pelo CNPQ (PDPG-TI, processo número 506555/06-4), que tem como principais objetivos o desenvolvimento de novos métodos e implementação de métodos já existentes de renderizações não fotorealísticas, criação de métodos que facilitem a interação com o usuário na geração de animações, além de investigar técnicas mais robustas e rápidas de segmentação e extração de movimentos em vídeos.

1.2 Contribuições

A principal contribuição desse trabalho é a criação de um método de renderização não fotorealística para estilização de imagens e vídeos, baseado na confecção artesanal das garrafas com areia colorida, o qual foi realizado através do desenvolvimento de uma aplicação que facilita a interação do usuário na criação e estilização das imagens e vídeos não fotorealísticos.

Para a estilização das imagens e vídeos usando areia colorida foi desenvolvida uma técnica para a geração de texturas procedurais de areia 2D, e duas técnicas que imitam os efeitos produzidos pelos artesãos, com sua ferramenta, na confecção das garrafas com areia colorida. É interessante de simular os efeitos produzidos pela ferramenta do artesão, quando o objetivo é produzir imagens ou vídeos bem similares aos tipos de imagens vistas em garrafas com areia colorida originais.

Contribuímos também com o desenvolvimento de um método para a criação de animações $2\frac{1}{2}$D, através da adição de informações 3D ao vídeo estilizado, onde os objetos que se movem no vídeo causam deslocamentos verticais nos grãos de areia da superfície plana de uma caixa de areia.

1.3 Organização do Trabalho

Esta dissertação é composta por um total de seis capítulos, onde os cinco capítulos restantes estão organizados da maneira a seguir:

- O Capítulo 2 trata sobre os métodos de renderizações não fotorealísticas de uma maneira geral, apresentando alguns trabalhos que foram importantes para a área, explicando algumas técnicas utilizadas e fazendo um breve histórico sobre a NPR. Outro ponto importante do capítulo é uma seção específica que trata sobre as técnicas aplicadas à estilização de vídeos, explicando sucintamente problemas existentes no processo de estilização do vídeo, como por exemplo, o efeito de *flickering* e o efeito *shower door*, e soluções para a coerência temporal das renderizações.

- O Capítulo 3 aborda sobre os métodos existentes utilizados para a realização do presente trabalho, explicando os motivos e como esses métodos foram utilizados. Este capítulo é de fundamental importância para a compreensão dos próximos capítulos. O capítulo descreve a segmentação *fuzzy* de vídeos, o modelo de cores HSV, a modelagem de campos de alturas e o algoritmo do mapa da distância.

- No Capítulo 4 é detalhado o método de renderização não fotorealística para estilização de imagens e vídeos proposto por este trabalho. Este capítulo é composto pela técnica de geração procedural de texturas, pelas técnicas que imitam os efeitos produzidos pelos artesãos através de sua ferramenta na produção das garrafas e pelo método para geração das animações $2\frac{1}{2}$D em caixas de areia, onde vários resultados com a utilização de vídeos reais e sintéticos são expostos através de figuras.

- O Capítulo 5 apresenta detalhes sobre: a implementação, a interface gráfica da ferramenta desenvolvida e como é realizada a interação da ferramenta com o usuário para a estilização de vídeos.

- No Capítulo 6 são feitas as considerações finais, concluindo o trabalho, e são expostas idéias para trabalhos futuros.

Capítulo 2

Renderização Não Fotorealística

Fotorealismo foi um estilo de pintura artística muito popular na América do Norte e Europa, onde artistas desenvolveram técnicas de pinturas a mão, que tentavam imitar imagens de câmeras fotográficas. Seus resultados eram tão bons que às vezes ficava difícil distinguir uma pintura artística de uma fotografia real. A partir daí, o termo fotorealismo foi designado para uma área da computação gráfica que tenta gerar imagens idênticas a fotografias reais, como os artistas daquela época, utilizando um conjunto de técnicas algorítmicas e as leis da física envolvidas no processo fotográfico [71].

A renderização fotorealística, desde a década de 60 e durante muitos anos, foi a mola propulsora das pesquisas na computação gráfica, tendo grandes avanços. Entretanto, em meados da década de 80, pesquisadores começaram a investigar como gerar imagens mais comunicativas, imagens com informações que não poderiam ser observadas em uma fotografia real. Segundo Foley (1990) [27], se o objetivo final de uma imagem é passar informação, então uma imagem livre de complicações como sombras e reflexos podem obter mais êxito do que uma imagem fotorealística. Essas informações só podiam ser vistas em imagens geradas por artistas como em ilustrações médicas e científicas, ilustrações técnicas, ilustrações arqueológicas, histórias em quadrinhos, etc. Com o objetivo similar ao fotorealismo, de gerar imagens através do computador, surge um novo campo de pesquisa em computação gráfica, chamado renderização não fotorealística (*Non Photorealistic Rendering* ou NPR), que tenta gerar imagens e animações através do computador com uma aparência de que foram feitas "a mão"[71].

Em 1963, Sutherland [73] publicou o primeiro artigo considerado como uma referência sobre NPR, onde desenvolveu um sistema chamado *Sketchpad*. O *Sketchpad* foi o progenitor de todos os sistemas de desenho e pintura computadorizados. Com a publicação de Sutherland [73], surge o paradigma de que o teclado não é a melhor forma de interação humano-computador. O paradigma de Sutherland é a base para as técnicas interativas.

O artigo de Appel et al. (1979) [1] foi considerado um dos primeiros trabalhos em NPR que empregam um *pipeline* que acessa diretamente uma estrutura de dados associada

a objetos 3D. Appel et al. [1] melhorou a renderização *wireframe* de objetos 3D pela introdução do estilo de ilustração técnica com um efeito de linha aureolar (quando uma linha da área visível do objeto se cruza com linhas da área escondida do objeto, parte das linhas de trás são apagadas na posição de encontro num formato aureolar, aumentando a sensação de profundidade).

Muitos artigos de renderização não fotorealística foram publicados antes mesmo da criação do termo renderização não fotorealística, como o de Sutherland [73] e o de Appel et al. [1]. O termo foi utilizado pela primeira vez por Lansdown e Schofield [45] em 1995. Os artigos que possuíam métodos de renderização muito brutos para serem classificados como fotorealísticas, passaram a ser considerados como de renderização não fotorealística, embora o objetivo deles fossem o fotorealismo. Entre os primeiros artigos publicados, o primeiro que se adequa ao que chamamos hoje de NPR, e que seu objetivo era a renderização não fotorealística é o artigo de Strassmann [70] em 1986. Neste artigo, Strassmann apresenta a idéia de "*path-and-stroke*", onde, inicialmente, um caminho é definido para que através da simulação física do pincel ou lápis, sejam geradas pinceladas ou riscos ao longo deste caminho.

A partir da década de 90, há uma explosão de artigos na área de Renderização Não Fotorealística. Dois artigos que tiveram bastante influência na NPR foram publicados em 1990, o de Saito e Takahashi [62] e o de Haeberli [34]. Saito e Takahashi [62] propõem métodos de renderização de imagens a partir de modelos 3D chamados de *Geometric-buffers* (*G-Buffers*) aumentando a compreensibilidade da renderização. Strothotte [71] define os *G-buffers* como a codificação de informações 3D em estruturas de dados 2D, que foram chamadas de estruturas de dados $2\frac{1}{2}$D. Os *G-buffers* consistem na obtenção de várias informações de um ambiente 3D através de imagens que podem ser obtidas por estes ambientes como as cores dos objetos da cena (*RGB-buffer*), identificadores de objetos (*ID-buffer*), informações de profundidades (z-*buffer*), informações sobre a superfícies das normais (n-*buffer*), informações codificadas nos materiais (*material-buffer*), e máscaras de sombras (*shadow-buffer*). Dessa forma, essas imagens armazenam informações sobre propriedades geométricas da cena e podem ser usadas para a renderização. Haeberli [34] apresenta um programa interativo que permite o usuário criar imagens com um estilo impressionista a partir de fotografias.

Outro artigo bastante importante para a NPR foi o de Salisbury et al. [63] em 1994. Neste artigo, são introduzidos algoritmos e estruturas de dados usados para a geração de ilustrações com caneta tinteiro (*pen-and-ink*) baseados no trabalho de Finkelstein e Salesin [26] denominado *Curvas de Multiresolução*. Salisbury et al. [63] criaram um sistema de ilustração 2D, onde texturas com padrões de riscos são utilizadas para fornecer textura e tonalidade às ilustrações. As ilustrações são criadas a partir de imagens guias que o usuário utiliza para aplicar as texturas. Neste artigo, Salisbury et al. [63] introduz a base dos algoritmos de diferenças de imagens para a seleção das posições das pinceladas.

Vários métodos que simulam estilos artísticos foram relatados na literatura como ilustrações técnicas [30, 32], ilustrações com caneta tinteiro [63, 80, 81], ilustrações com grafite [67, 69, 68], pintura em aquarela [8, 19], pintura impressionista [48, 53], mosaicos [21, 22, 24], *cartoons* [44, 76], vitrais [54], dentre outros. Alguns exemplos de renderizações não fotorealísticas são mostrados na Figura 2.1.

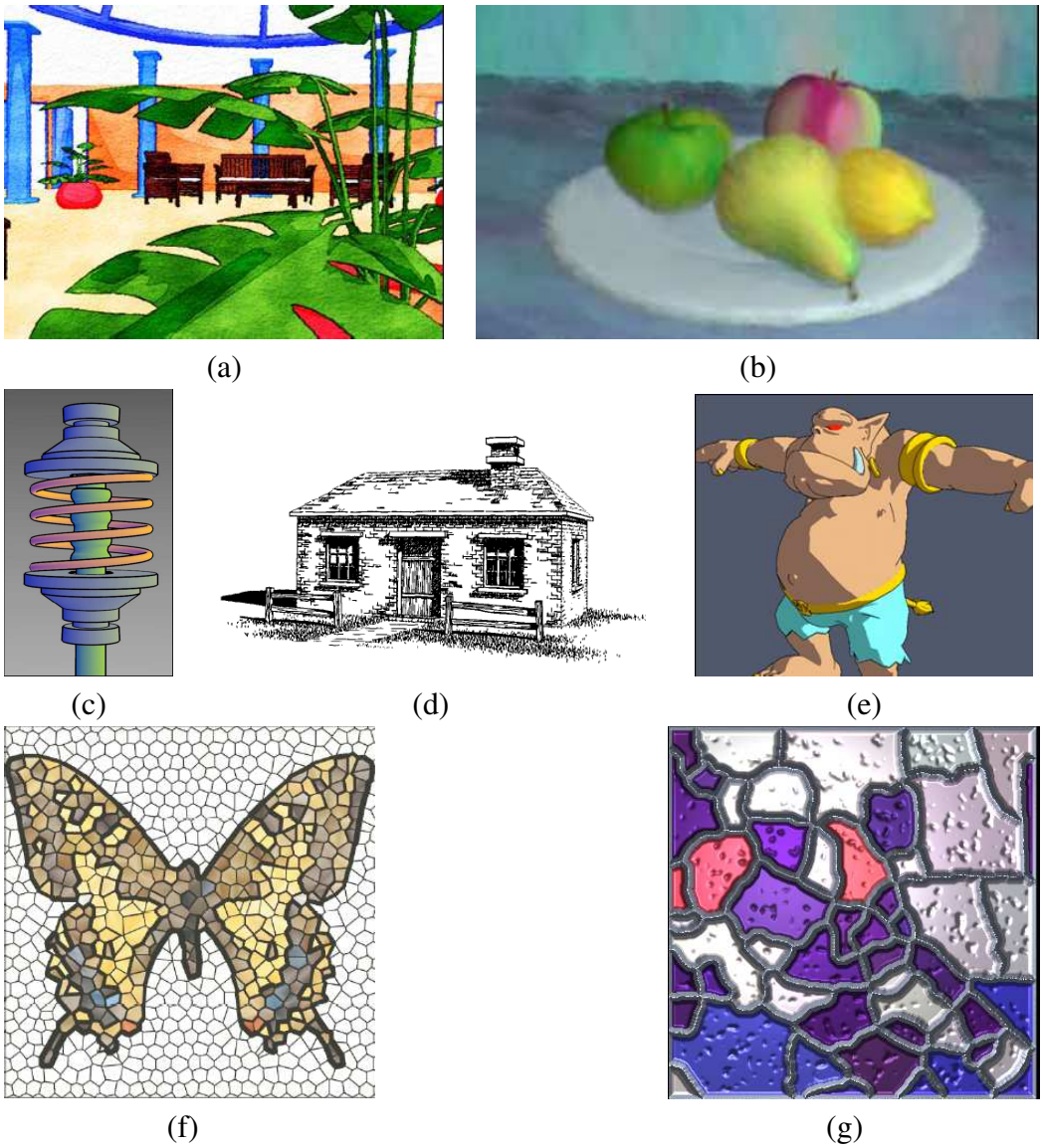


Figura 2.1: Exemplos de trabalhos em NPR: (a) pintura em aquarela (2006) Bousseau et al. [8], (b) pintura em pastel (1996) Meier [53], (c) ilustração técnica (1998) Gooch et al. [30], (d) ilustração com caneta tinteiro (1994) Winkenbach e Salesin [80], (e) *cartoon* (2000) Lake et al. [44], (f) mosaico (2002) Dobashi et al. [22] e (g) vitral (2003) Mould [54].

Devido à vasta quantidade de trabalhos e a diversidade de estilos na NPR, surgiu a necessidade de organizar os trabalhos, agrupando-os em categorias com propriedades e características em comum, para que os estudos sobre NPR ficassem mais didáticos. Dessa forma, Gooch e Gooch [31] propuseram que as pesquisas em NPR fossem classificadas

em três categorias gerais:

- *Simulação do meio artístico*: representam as técnicas que tentam criar uma aproximação das propriedades físicas do meio artístico (e.g. pintura com aquarela, ilustração com caneta tinteiro, ilustração com grafite);

- *Criação de imagem com auxílio do usuário*: representam métodos que utilizam softwares para guiar o usuário na produção de imagens artísticas. Pesquisas neste formato incorporam nos softwares desenvolvidos habilidades e técnicas artísticas, para que usuários sem conhecimento artístico possam produzir imagens que transmitam uma sensação de que foram feitos a mão;

- *Criação automática de imagens*: representam técnicas que tentam criar automaticamente imagens artísticas de acordo com uma especificação prévia de objetivos.

Sousa [66] propôs um esquema de classificação mais completo, que classifica as técnicas desenvolvidas em quatro categorias que correspondem a aspectos importantes da NPR. As técnicas podem ser analisadas sobre todas as quatro categorias, que são:

- *Material*: a simulação de meio natural para desenho (lápis, carvão, caneta tinteiro, grafite, papel), pintura (aquarela, óleo), ferramentas ou primitivas de renderização (pincel, quadro, pincelada, borracha);

- *Estrutura*: as fontes de entrada para renderização podem ser livre de composição, renderização direta em 3D, renderização direta em $2\frac{1}{2}$D, ou uma composição destas. Renderização direta em 3D são os métodos que acessam diretamente estrutura de dados e algoritmos de uma cena 3D. Já na renderização direta em $2\frac{1}{2}$D (como Strothotte [71] denominou este tipo de acesso), as cenas 3D são previamente processadas, adquirindo um conjunto de informações 2D que são armazenadas em *buffers* para depois serem processadas por algoritmos de NPR.

- *Método*: regras são usadas para criar a imagem, que podem ser geradas pela interação do usuário ou de forma automática;

- *Aplicação*: a intenção do usuário com o sistema e o contexto da aplicação correspondente. O público alvo determina o estilo apropriado de renderização (ilustração técnica, *cartoon*, pinturas, ilustrações em geral) para o contexto de aplicação desejado (arte, visualização científica, ilustração médica, arquitetura, designer).

2.1 Estilização de Vídeos

No início das pesquisas sobre NPR, muitos trabalhos desenvolveram técnicas para a renderização estilizada de imagens ou a renderização de cenas 3D estáticas. Em 1996, Meier

[53] apresenta o primeiro trabalho relacionado a renderização não fotorealísticas a partir de animações em cenas 3D. Logo em seguida, Litwinowicz (1997) [48] apresenta o primeiro trabalho propondo o uso de renderizações estilizadas para o processamento de vídeos.

Um dos objetivos das técnicas de NPR para estilização de vídeos é a automatização ou semi-automatização de tarefas que imitam estilos artísticos, permitindo que usuários estilizem vídeos capturados por uma câmera, gerando animações com poucos esforços em relação à criação de animações no modo tradicional. A estilização de vídeo também oferece a possibilidade de misturar vídeos reais com objetos estilizados, renderizando os objetos com uma ou mais técnicas NPR, além de poder estilizar somente partes dos vídeos.

Um dos fatores mais importantes na produção de vídeos estilizados, seja a partir da descrição de uma cena artificial 3D ou de um vídeo real, é a exibição de coerência temporal dos elementos de desenho (e.g. pinceladas, curvas, círculos, etc). Isso é feito movendo-se os elementos de desenho de acordo com a superfície dos objetos que estão sendo desenhadas, no caso das cenas 3D. Caso isto não seja feito, a animação parecerá como sendo vista através de um vidro texturizado, também chamado de efeito *shower door* [53].

A falta de coerência temporal na renderização estilizada de vídeos reais pode causar oscilações e/ou tremulações chamados de *flickering*. O efeito de *flickering*, também conhecido como *swimming* [16], surge tanto em objetos que estão em movimento e são renderizados com elementos que não os seguem corretamente quanto em áreas estáticas que são renderizadas diferentemente nos quadros adjacentes, devido a ruídos na captura dos vídeos ou algumas diferenças de sombreamento e iluminação.

Várias abordagens tentam impor a coerência temporal na estilização de vídeos. No trabalho de Litwinowicz [48], foi introduzido um método para manter a coerência temporal em pinturas impressionistas. Para isto, foram utilizadas informações de um método de fluxo óptico [20] para rastrear movimentos no vídeo, e desta forma poder movimentar, adicionar ou remover pinceladas de um quadro para outro. Um método para a renderização coerente de áreas estáticas em quadros sucessivos foi proposta por Hertzmann e Perlin em [38], onde são mantidas as pinceladas das áreas que não apresentam movimentos. A coerência temporal é obtida através do deslocamento de pontos de controles das pinceladas usando informações de um método de fluxo óptico.

No trabalho de Wang et al. [76], os autores propõem um método para criar *cartoons* a partir de vídeo utilizando um algoritmo de segmentação para segmentar o vídeo por completo. Após a segmentação, o usuário especifica pontos em quadros chaves do vídeo através de uma interface gráfica. Esses pontos são então usados para interpolar as bordas das regiões entre os quadros.

No método proposto por Collomosse et al. [16], o vídeo é tratado como um volume

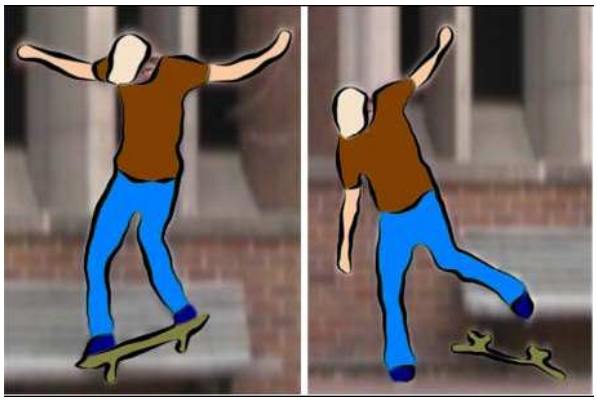
3D $I(x, y, z)$, onde x é a largura dos quadros do vídeo, y é a altura dos quadros e z é cada quadro do vídeo. Neste método é aplicado um algoritmo de segmentação 2D em cada quadro do vídeo seguido de um algoritmo que, baseando-se em regras heurísticas, associa as regiões segmentadas entre os quadros. O resultado deste método são objetos temporalmente convexos segmentados forçando a coerência temporal na renderização dos mesmos.

Winnemöller [82] fornece um certo nível de coerência temporal a vídeos estilizados como *cartoons*, através de uma quantização suave das cores dos quadros abstraídos. A quantização suave das cores faz com que mudanças de iluminação causadas na captura do vídeo sejam espalhadas pelo vídeo diminuindo a notoriedade dessas mudanças.

Em Gomes et al. [28], é apresentado um método para forçar a coerência temporal intra-objeto em NPR de vídeos utilizando a combinação de um algoritmo de segmentação [14] e um algoritmo de fluxo óptico [4]. O resultado do algoritmo de segmentação é utilizado para restringir a área de busca do algoritmo de fluxo óptico para as bordas do objeto segmentado. A restrição da área na qual o fluxo óptico é calculado possibilita a obtenção de mapas de fluxos mais precisos. Desse modo, a informação de fluxo óptico pode ser utilizada para forçar a coerência temporal nestes vídeos.

A Figura 2.2 mostra alguns exemplos de trabalhos em NPR sobre estilização de vídeos.

(a)



(b)

(c)

Figura 2.2: Exemplos de trabalhos em NPR sobre estilização de vídeo: (a) (2005) Collomosse et al. [16], (b) (2005) Wang et al. [75], (c) (2006) Gomes et al. [28].

Capítulo 3

Fundamentos e Métodos Utilizados

Para a implementação desse trabalho foi necessário o uso de alguns métodos existentes como: a segmentação *fuzzy* de vídeos [14], o modelo de cores HSV [64], a modelagem de campos de alturas e o algoritmo do mapa de distância. A seguir, será explicado sucintamente o funcionamento de cada um desses métodos e como eles foram utilizados no presente trabalho.

3.1 Segmentação *Fuzzy* de Vídeos

Muitas técnicas de segmentação de vídeos têm sido utilizadas em aplicações para o armazenamento, recuperação e reconhecimento de informações existentes em vídeos [9], na criação de animações e renderizações não fotorealísticas [12, 76], em sistemas para compressão de vídeos [15] ou em sistemas de monitoramento de tráfego de automóveis [74, 6]. Essas técnicas têm sido desenvolvidas com a intenção de superar as dificuldades inerentes à segmentação de vídeos, e atender as novas necessidades dos sistemas que utilizam a segmentação em uma de suas etapas de funcionamento.

A segmentação de um vídeo apresenta diversos problemas comuns na segmentação de imagens como o tratamento de imagens corrompidas por ruídos, iluminação não uniforme e *background* irregular, além das dificuldades particulares, como segmentar vídeos contendo objetos não rígidos e movimentos bruscos. A segmentação de vídeo permite adicionar informações de alto nível ao processo de renderização, de modo que se o vídeo não está bem segmentado, a sua renderização também perderá qualidade. Esta perda de qualidade ocorre devido a alguns pixels serem classificados de forma errada como pertencentes a um objeto, e os seus correspondentes no próximo quadro forem classificados como pertencentes a outro objeto. Dessa forma, a renderização do vídeo não terá uma coerência temporal. Esse efeito indesejável é geralmente observado nas bordas de objetos e acentuado em vídeos de baixa resolução e com ruídos.

A segmentação *fuzzy* é uma técnica que determina para cada pixel da imagem um grau

de pertinência, no intervalo [0,1], indicando a certeza desse pixel pertencer ou não a um determinado objeto existente na imagem [13]. Com essas informações de pertinência, o algoritmo de segmentação pode tomar uma decisão mais flexível sobre a classificação de cada pixel da imagem. A aplicação do algoritmo de segmentação *fuzzy* em vídeo, trata-o como um volume 3D ($I(x, y, z)$, sendo z cada quadro do vídeo), onde para cada voxel (um pixel no espaço 3D), é calculado o grau de pertinência, gerando um mapa $M(x, y, z)$ contendo o grau de pertinência de todos os voxels do volume 3D. Este mapa é chamado de *mapa de conectividade*.

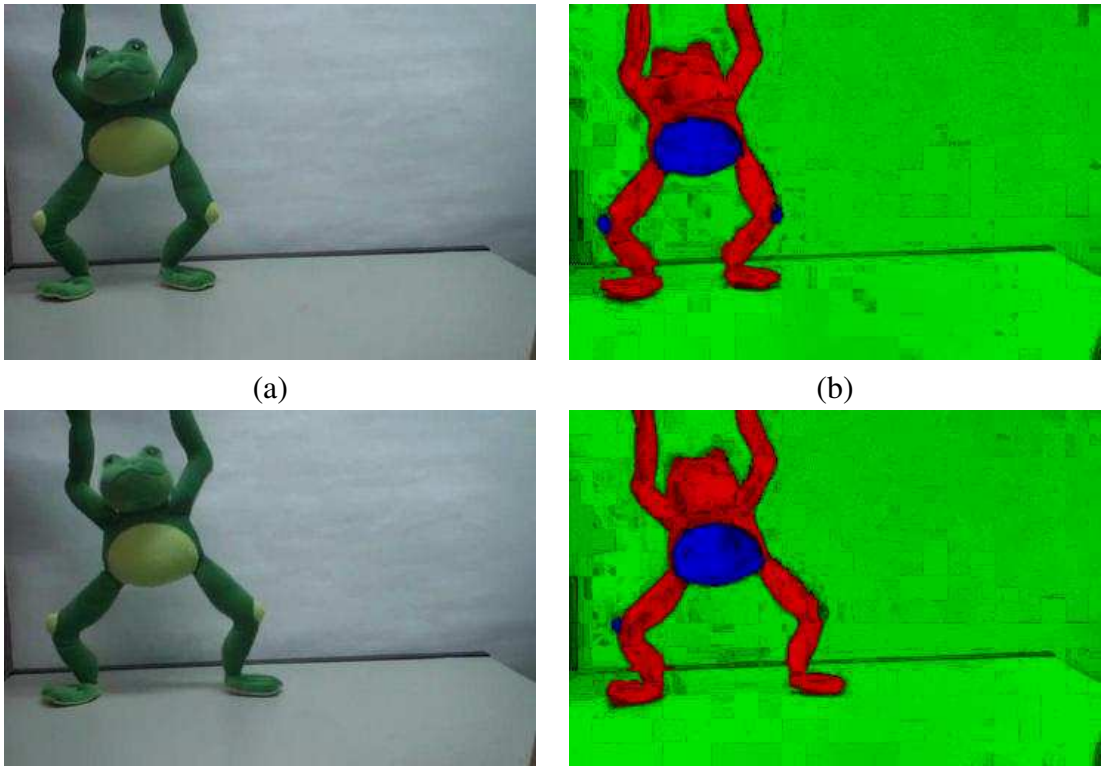
Além de informações de cor para a segmentação *fuzzy* de vídeos, a segmentação *fuzzy* pode utilizar informações de movimento para obter uma segmentação mais robusta como proposto por Khan e Shah em [43]. Devido a alguns vídeos utilizados na segmentação apresentarem resultados insatisfatórios quando a cor do objeto a ser segmentado é parecida com o *background*, as informações de movimento podem auxiliar na distinção dos objetos a serem segmentados. Entretanto, quando utiliza-se somente informações de movimento na segmentação podem ocorrer problemas, devido à oclusão de objetos e à iluminação não uniforme, a estimação do movimento pode apresentar imprecisão nas bordas e em algumas partes dos objetos em movimento.

As estimações de movimento podem ser obtidas através de algoritmos de fluxo óptico [59, 4]. De acordo com Horn e Schunck [39], o fluxo óptico é a distribuição aparente de velocidade que um padrão de intensidade apresenta quando se move em uma imagem, ou seja, é um método para estimação de movimentos que calcula uma aproximação de movimento baseando-se na derivada local de uma dada seqüência de imagens. No caso do vídeo, ele especificará quanto cada pixel se moveu de um quadro para o outro.

A utilização de lógica *fuzzy* no algoritmo de segmentação permite que se renderize objetos usando diferentes estilos, de acordo com o seu grau de pertinência. Dentro de cada objeto segmentado há pequenos detalhes da imagem relacionados com a variação dos graus de pertinência do próprio objeto que poderão ser renderizados de formas diferentes dentro de um mesmo objeto, como pode ser visto os detalhes nos mapas de segmentação da Figura 3.1 (b) e (d).

O algoritmo utilizado é um método semi-automático de segmentação por crescimento de regiões, onde, de forma interativa, o usuário seleciona um ou mais pixels sementes para representar os objetos que deseja segmentar. Neste trabalho, utilizou-se uma variante do algoritmo rápido de segmentação *fuzzy* (MOFS) introduzido por Carvalho et al. em [13], e estendido para segmentar volumes 3D coloridos (vídeos) por Carvalho et al. em [14]. Além disso, esta segmentação utiliza um algoritmo de fluxo óptico para estimação de movimentos que implementa o método diferencial da implementação multi-resolução do algoritmo de Proesmans [59] desenvolvido por McCane et al. em [51]. Todos os vídeos, aqui presentes, foram segmentados usando-se o módulo de segmentação do programa *AVP Rendering* desenvolvido pelo grupo *AnimVideo*.

A Figura 3.1 mostra um exemplo de segmentação *fuzzy* de um vídeo onde um sapo de pelúcia anda da esquerda para a direita. O exemplo a seguir, mostra o oitavo e o vigésimo quadro do vídeo, onde foram segmentados três objetos da seguinte forma: de azul a barriga e os joelhos do sapo, de vermelho o restante do corpo do sapo e de verde o chão e a parede.



(a) (b)

(c) (d)

Figura 3.1: Dois quadros do vídeo do sapo em (a) e (c), os mapas de segmentação *fuzzy* dos quadros em (b) e (d) respectivamente. (2007) Oliveira [57].

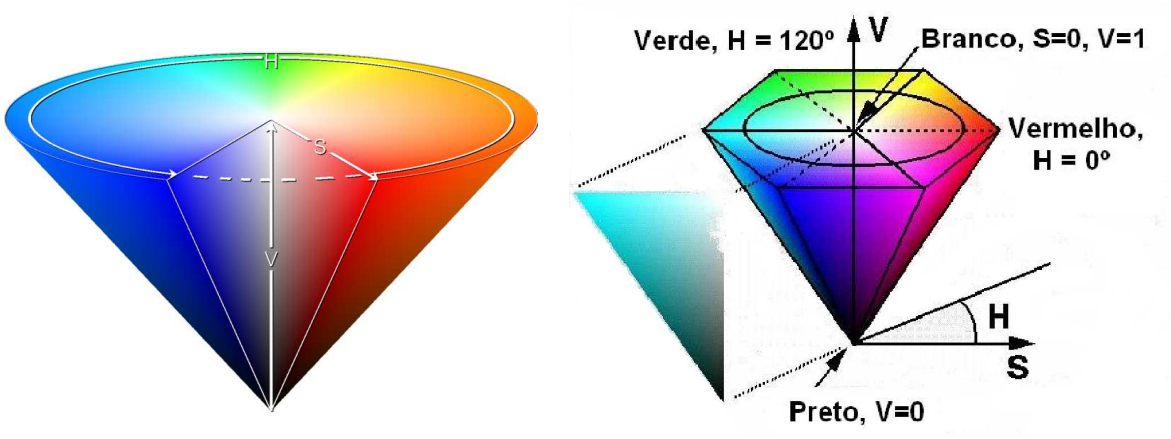
Segundo Litwinowicz [48], a aplicação de técnicas de renderização não fotorealísticas para estilização de vídeos geralmente é dividida na segmentação do vídeo para a extração de objetos de interesse, seguida da renderização não fotorealística das regiões de interesse. Neste trabalho, utilizou-se a segmentação *fuzzy* [14] para a extração dos objetos de interesse do vídeo, devido sua robustez na presença de ruídos, rapidez e flexibilidade.

3.2 Modelo de Cores HSV

Um modelo de cores é uma especificação de um sistema de coordenadas 3D de cores e de um subconjunto neste sistema de coordenadas, onde todas as cores visíveis estão em uma posição particular do subconjunto [27]. Em outra palavras, é um método que explica as propriedades ou comportamento da cor. Por exemplo, o modelo de cores RGB (*Red, Green, Blue*), um dos modelos mais conhecido por ser usado em monitores, é formado por

um sistema de coordenadas cartesianas 3D, onde é definido um cubo como subconjunto visível. Existem vários modelos de cor, uns orientados a hardware e outros orientados a usuário. O RGB é um modelo orientado a hardware, pois foi definido pela cromaticidade do fósforo usados nos monitores CRT (*Cathode Ray Tube*). Nenhum modelo simples pode explicar todos os aspectos da cor, mas é possível usar diferentes modelos para ajudar na descrição das diferentes características da cor [36].

O modelo de cores HSV (*Hue, Saturation, Value* - Matiz, Saturação, Valor), também chamado de HSB (B de *brightness*, brilho), foi desenvolvido por Smith em [64]. É um modelo de cores orientado a usuário, pois se baseia na maneira com que um artista descreve e mistura as cores. O sistema de coordenadas é um cone (ver Figura 3.2(a)), e o subconjunto dentro do qual o modelo é definido é um hexacone, ou uma pirâmide de seis lados (ver Figura 3.2(b)).



(a) (b)

Figura 3.2: O Modelo de Cores HSV.

O matiz, ou H, é dado pelo ângulo ao redor do eixo vertical, variando no intervalo de 0°, cor vermelha, até 360° que é igual a 0°. Seus vértices são separados em intervalos de 60°, sendo o amarelo igual a 60°, o verde igual a 120° e assim por diante.

No topo do hexacone, onde $V = 1$, são encontradas as cores com maior intensidade. O eixo vertical V representa a escala de tons de cinza (*grayscale*), que varia no intervalo contínuo de $[0, 1]$, onde sempre que $V = 0$ sua cor é preta não importando o valor de S e H. A saturação, ou S, é usada para determinar a pureza da cor, variando também em um intervalo contínuo de $[0, 1]$, onde 0 fica na linha do eixo vertical (no centro do hexacone) e vai até 1 nos lados do hexacone. Quando $S = 0$, não é definido um valor para o matiz H, dessa forma a cor vai depender somente do valor de V, ou seja, vai estar na escala de tons de cinza. Se $V = 1$ e $S = 0$ a cor é branca. As cores quando os valores de V e S são iguais a 1, são ditas de matiz pura, assemelhando-se aos pigmentos usados por artistas.

O sistema HSV utiliza descrições de cor que são mais intuitivas do que as combinações de cores primárias como o RGB, e por isso, é mais adequado para utilização por

usuários na especificação de cores em interfaces gráficas [3]. Sendo $H \in [0^o, 360^o]$, $S, V, R, G, B \in [0, 1]$, MAX igual ao máximo valor entre (R, G, B) e MIN o mínimo, as conversões entre os modelos de cor RGB e HSV são dadas por:

- de RGB para HSV

$$
\begin{aligned}
H &= \begin{cases} indefinido, & se\ MAX = MIN \\ 60^o \times \frac{G-B}{MAX-MIN} + 0^o, & se\ MAX = R\ e\ G \geq B \\ 60^o \times \frac{G-B}{MAX-MIN} + 360^o & se\ MAX = R\ e\ G < B \\ 60^o \times \frac{B-R}{MAX-MIN} + 120^o, & se\ MAX = G \\ 60^o \times \frac{R-G}{MAX-MIN} + 240^o, & se\ MAX = B \end{cases} \\
S &= \begin{cases} 0, & se\ MAX = 0 \\ 1 - \frac{MIN}{MAX}, & caso\ contrário \end{cases} \\
V &= MAX
\end{aligned}
\tag{3.1}
$$

- de HSV para RGB

$$
\begin{aligned}
& H_i = \left\lfloor \tfrac{H}{60} \right\rfloor mod\ 6 \\
& f = \tfrac{H}{60} - H_i \\
& p = V(1 - S) \\
& q = V(1 - fS) \\
& r = V(1 - (1 - f)S \\
& se\ H_i = 0 \longrightarrow R = V, G = t, B = p \\
& se\ H_i = 1 \longrightarrow R = q, G = V, B = p \\
& se\ H_i = 2 \longrightarrow R = p, G = V, B = t \\
& se\ H_i = 3 \longrightarrow R = p, G = q, B = V \\
& se\ H_i = 4 \longrightarrow R = t, G = p, B = V \\
& se\ H_i = 5 \longrightarrow R = V, G = p, B = q
\end{aligned}
\tag{3.2}
$$

Na computação gráfica, algumas vezes os valores dos componentes dos modelos HSV e RGB são representados no intervalo de $[0, 255]$ ao invés de números ponto flutuante entre 0 e 1. Neste caso, as conversões não cobrem todos os pontos do espaço e algumas distorções são causadas por arredondamento.

O modelo de cores HSV foi escolhido para ser usado neste trabalho por ser um modelo orientando a usuário, sendo, desta forma, um modelo que fornece uma maior facilidade na seleção de cores. Além destes, existem outros motivos para a escolha deste modelo que serão explicados no próximo capítulo, no momento da descrição do método de renderização proposto.

3.3 Campos de Alturas

Segundo Zhao [85], *campos de alturas* são matrizes de duas dimensões, usadas para armazenar dados que representam alturas de um tipo de terreno (e.g. água, areia, grama, etc) em um ponto (x, y) especifico do espaço. Também é comumente utilizado no cálculo de *bump maps*, que é uma técnica utilizada para dar mais realismo a imagens sintéticas, aplicando uma função de perturbação nas normais para a utilização do vetor normal perturbado no cálculo do modelo de iluminação [36]. Campo de alturas foi primeiramente proposto para uso em *video games*, sendo usado como terrenos sobre os quais os personagens andam e saem vasculhando pelo mundo virtual. Para renderizar um campo de alturas é necessário gerar uma malha poliédrica. Nos triângulos formados pela malha são aplicadas texturas do tipo de material que será utilizado para simular a superfície. Os vértices dos triângulos funcionam como as colunas do campo de altura e podem ser deslocadas verticalmente. A Figura 3.3 mostra um exemplo da aplicação do campo de alturas para a simulação de terrenos produzido por McGuire e Sibley [52].

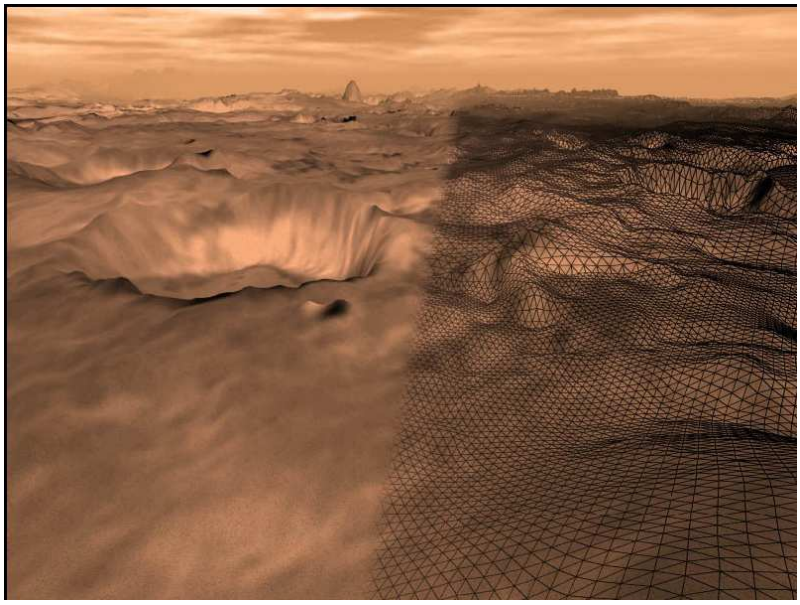


Figura 3.3: Exemplo de campo de alturas simulando terrenos. (2005) McGuire e Sibley [52].

Na renderização do campo de alturas é necessário calcular as normais da malha poliédrica. Existem vários algoritmos para o cálculo das normais de malhas poliédricas. Estes algoritmos tem diferenças substanciais, mas todos eles tem em comum a noção de dar pesos de algum modo às normais das faces adjacentes, que é dado pela fórmula

$$\mathbf{n} = \sum_{i=1}^{m} w_i n_i, \tag{3.3}$$

onde é feito o somatório de todas as m faces da malha poliédrica que contêm o vértice em

questão, n_i é a normal da face, e w_i é um peso que pode ser um ângulo ou o tamanho de uma aresta [85]. Para o cálculo das nomais de cada vértice, foi utlizado o algoritmo do *Mean Weighted Equally* (MWE) introduzido por Gouraud [33]. Neste algoritmo, $w_i = 1$ fazendo com que a normal de cada face contribua igualmente para o cálculo da normal do vértice. Em seguida é necessário um passo trivial de normalização [41], onde é calculada a média do somatório.

Para o cálculo das normais dos vértices de uma malha poliédrica, é necessário o cálculo das normais das faces. Um vertor normal n em um ponto P de uma superfície S é um vertor perpendicular à superfície S no ponto P. O vetor normal da face geralmente é calculcado através do produto cartesiano de duas arestas do triângulo, que é dado pela fómula

$$n = \frac{(v_1 - v_0) \times (v_2 - v_0)}{||(v_1 - v_0) \times (v_2 - v_0)||}, \tag{3.4}$$

onde v_0, v_1 e v_2 são os vértices do triângulo e cada vértice $v(x, y, z)$ pertence a um ponto no espaço [42]. A divisão na fórmula é feita para a normalização do vetor normal n.

O campo de alturas foi utilizado para adicionar ao vídeo estilizado informações 3D. Cada pixel de um quadro do vídeo estilizado é mapeado em cada vértice dos triângulos da malha poliédrica, que são deslocados verticalmente para a geração das animações $2\frac{1}{2}$D. A seguir, o Algoritmo 1 mostra o pseudo-código OpenGL [55] utilizado para renderizar o campo de alturas.

Algoritmo 1: Pseudo-código OpenGL para renderização do campo de alturas

```
for (int j = 0; j < video.height()-1; j++) do
    glBegin(GL_TRIANGLE_STRIP);
    for (int i = 0; i <video.width(); i++) do
        glNormal3fv(heightfield[j][i].normal);
        glColor4fv(heightfield[j][i].color);
        glVertex3fv(heightfield[j][i].position);
        glNormal3fv(heightfield[j+1][i].normal);
        glColor4fv(heightfield[j+1][i].color);
        glVertex3fv(heightfield[j+1][i].position);
    end
    glEnd();
end
```

3.4 Mapas de Distância

Uma *transformada de distância*, também conhecido como mapa de distância, campo de distância ou mapa de contorno, é uma ferramenta computacional usada para extrair infor-

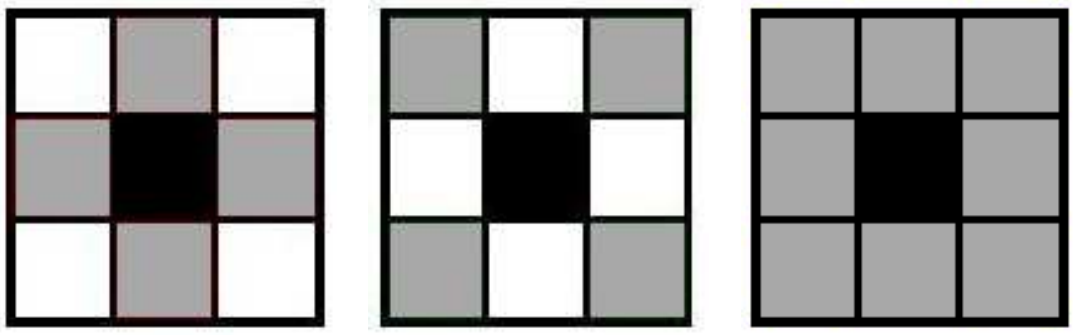
mações sobre a forma e a posição dos pixels de uma região de interesse da imagem em relação aos outros pixels [46]. A tranformada de distância produz um mapa de distância a partir de uma imagem binária. Para cada pixel dentro do objeto da imagem binária, o pixel correpondente no mapa de distância tem o valor igual à mínima distância deste pixel em relação ao *background*. Estas transformadas são usadas extensivamente no campo da visão computacional e processamento de imagens, como por exemplo na reconstrução de objetos através de partes de bordas [50], esqueletonização [2] e cálculo de diagramas de Voronoi [83].

Existem vários tipos diferentes de transformadas de distância que são baseadas nos diferentes tipos de medidas de distância. Dentre as medidas estão a distância Euclidiana, a distância *city-block* (também conhecida como distância de *Manhattan*) e a distância *chessboard*. A distância Euclidiana entre os pixel p e q, com coordenadas (x, y) e (s, t) respectivamente, é definida por

$$D_e(p, q) = \sqrt{(x - s)^2 + (y - t)^2}. \tag{3.5}$$

Segundo Gonzalez e Woods [29], para esta medida de distância, os pixels tem uma distância menor ou igual a algum valor r de (x, y) que são pontos contidos em uma circunferência de raio r centrada em (x, y).

A distância *city-block* e a distância *chessboard* são baseadas em vizinhanças. Ao redor de um pixel existem 8 pixels vizinhos, sem considerar os pixels que se localizam nas bordas da imagem. Existem vários tipos de vizinhanças entre pixels, como a vizinhança 4, a vizinhança D (diagonal) e a vizinhança 8. A Figura 3.4 apresenta estes três tipos de vizinhança, onde em (a) o algoritmo só considera os vizinhos na horizontal e vertical, em (b) só os vizinhos das diagonais e em (c) todos os 8 vizinhos.



(a) (b) (c)

Figura 3.4: Vizinhança 4 (a), vizinhança D (b) e vizinhança 8 (c).

Logo o conjunto de vizinhos com vizinhança 4 (V_4) do pixel p com as coordenadas (x, y) é dado por

$$V_4(p) = \{(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)\}, \tag{3.6}$$

o conjunto de vizinhos com vizinhança D (V_D) do pixel p é dado por

$$V_D(p) = \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\} \tag{3.7}$$

e o conjunto de vizinhos com vizinhança 8 (V_8) é a união dos conjutos (V_4) e (V_D).

$$V_8(p) = V_4(p) \cup V_D(p) \tag{3.8}$$

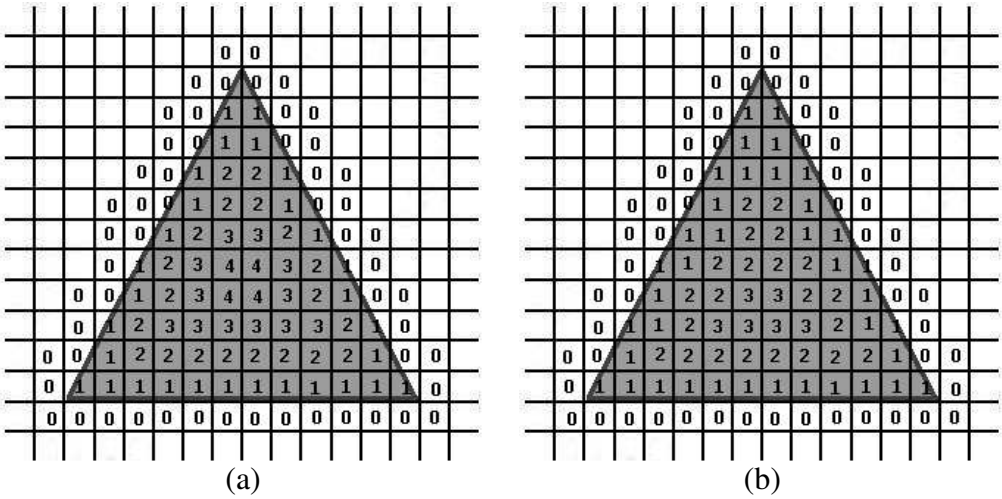
O algoritmo para cálculo do mapa de distância percorre uma imagem e em cada pixel verifica se seus vizinhos fazem parte do objeto de interesse. Se não fizer parte, aquela posição recebe o valor zero, e se fizer parte o valor da posição vai sendo incrementado de acordo com a sua distância em relação à borda. Dessa forma, o algoritmo pode considerar apenas alguns vizinhos dependendo da medida de distância escolhida. A medida de distância *city-block* usa a vizinhança 4. A distância *city-block* entre um pixel p e q, com coordenadas (x, y) e (s, t) respectivamente, é definida por

$$D_4(p, q) = |x - s| + |y - t|. \tag{3.9}$$

A medida de distância *chessboard* usa a vizinhança 8 é definida por

$$D_8(p, q) = max(|x - s|, |y - t|). \tag{3.10}$$

A escolha da medida de distância depende da finalidade que o algoritmo será utilizado. No caso desta simulação foi utilizado o mapa de distância *chessboard*, pois os resultados ficaram mais suavizados em relação ao mapa de distância *city-block*. O algoritmo do mapa de distância foi utilizado em várias técnicas desenvolvidas nesse trabalho. A Figura 3.5 mostra os resultados obtidos com o cálculo do algoritmo do mapa de distância *city-block* e *chessboard*.



(a) (b)

Figura 3.5: Resultado do mapa de distância *city-block* (a), e resultado do mapa de distância *chessboard* (b).

Capítulo 4

Renderizações Não Fotorealísticas com Areia Colorida

A criação de imagens de paisagens com o preenchimento de garrafas com areia colorida é uma expressão artística desenvolvida por famílias de artesãos do litoral do Nordeste do Brasil, principalmente nos estados do Rio Grande do Norte e Ceará. No litoral desses estados, existem areias com várias cores e tonalidades diferentes, que podem ser encontradas em praias e dunas. Um exemplo típico de garrafa preenchida com areia colorida produzida utilizando esta técnica pode ser vista na Figura 4.1.

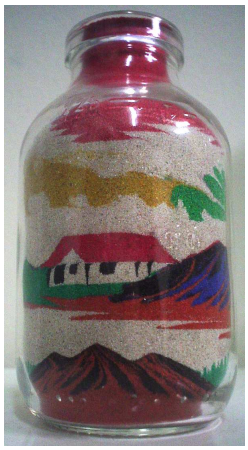
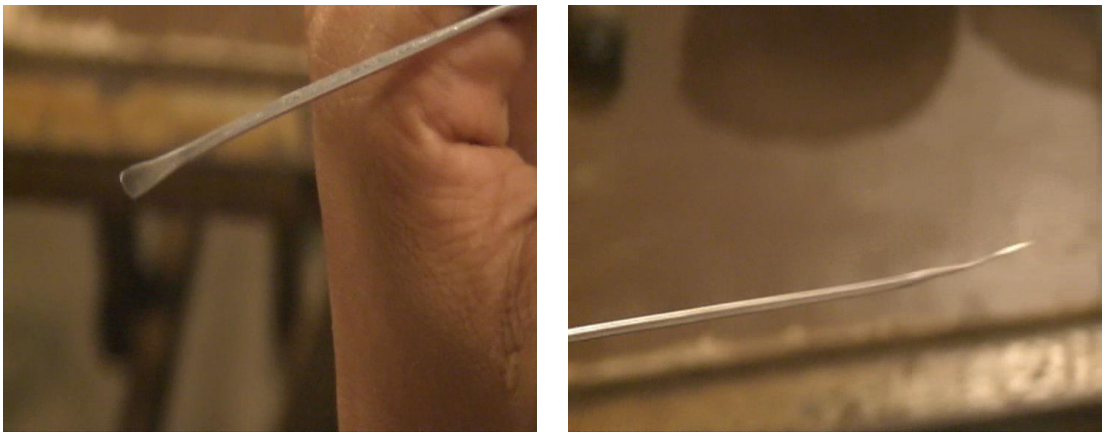


Figura 4.1: Garrafa produzida pelo preenchimento de areia com diferentes cores de forma ordenada.

A técnica para fazer tais peças de arte consiste no preenchimento de uma garrafa de vidro vazia com areia de diferentes cores, movendo-as para formar uma imagem usando um fino pedaço de madeira, ou mais recentemente, uma ferramenta de ferro onde uma das pontas é fina e aguda e a outra é chata parecida com uma chave de fenda, como pode ser vista na Figura 4.2. Depois de preencher a garrafa com uma quantidade de areia, o artesão usa a ponta chata da ferramenta para remover excessos de areia de alguma cor da superfície interna da garrafa para o centro dela, delineando as bordas de objetos que

estão sendo desenhados. Então, o artesão usa a ponta fina da ferramenta para refinar as formas dos objetos e uma das duas pontas para criar detalhes dos objetos empurrando ou puxando areia de diferentes cores das áreas envolvidas. Por exemplo, ao desenhar uma casa, o artesão primeiramente compõe as paredes visíveis com areia, derrama areia de uma cor diferente para as portas e janela em cima da areia da parede, e então ele empurra a areia das portas e janelas de forma que na superfície interna do vidro elas sobreponham a areia da parede, formando assim a imagem da porta e da janela. Finalmente o artesão derrama areia para o telhado organizando-a para terminar a casa. Uma variação desta arte é a criação de imagens entre duas peças chatas de vidro, produzindo uma "pintura"que pode ser colocada em uma superfície plana.



(a) (b)

Figura 4.2: A ponta chata (a) e a ponta aguda (b) da ferramenta utilizada pelos artesãos para compor as imagens dentro das garrafas.

As areias naturais originalmente utilizadas para criar essas imagens dentro das garrafas eram extraídas das áreas litorâneas que agora estão protegidas por questões ambientais, principalmente para evitar a degradação desses locais causados pela coleta de areia. (Alguns deles são agora atrações turísticas.) Assim, hoje em dia, os artesãos foram obrigados a trocar as areias naturais por areias artificiais, o que se mostrou vantajoso, já que são mais fáceis de se manipular, e que podem ser encontradas em várias cores que não eram disponíveis na natureza, como azul, verde e roxo.

A simulação de tais imagens usando a renderização estilizada também possibilita a produção de algo que é quase impossível na vida real: a criação de animações estilizadas de areia e vídeos com a técnica artística descrita acima. Para isto, um artesão teria que criar uma série de garrafas ou "pinturas", com a mesma paisagem de fundo e com alguns objetos se movendo. É fácil ver que fazer isto é extremamente difícil, e que uma animação gerada dessa maneira provavelmente teria que ser muito simples para permitir a criação de tais seqüências.

Existem outros trabalhos com areia, mas que objetivam o fotorealismo. Hanrahan e Krueger (1993) [35] descrevem um modelo de reflexão de superfícies que possuem ca-

madas, tais como tecidos biológicos (e.g. pele, folha, etc.) ou materiais inorgânicos (e.g. areia, neve, tinta, etc.), causada pela dispersão da luz através das camadas da superfície. Sumner et al. (1999) [72] introduzem um modelo para deformar materiais de terrenos como areia, lama e neve, onde objetos rígidos em 3D se movimentam sobre estes terrenos deixando seus rastros ou pegadas. Zhu e Bridson (2005) [86] descrevem um método baseado em física para animar areia, onde é abstraído os grãos individuais da areia em um objeto contínuo, e através de pequenas mudanças em um simulador de água existente é gerado um simulador de areia. Bell et al. (2005) [5] apresentam um método que simula materiais granular, tais como areia e grãos, onde partículas representam elementos discreto do material simulado e fenômenos altamente dinâmicos como avalanches e esparrames são gerados.

4.1 Texturas Procedurais de Areia

A geração de texturas procedurais é uma técnica bastante usada na simulação de materiais naturais. Geralmente são utilizadas funções de ruído (*noise*) ou turbulência (*turbulence*). Por exemplo, o *Perlin Noise* [58] é uma técnica que usa uma função ruído para gerar proceduralmente texturas 3D que simulam mármore, pedra e madeira [23]. No caso deste trabalho, utilizou-se uma função de ruído para gerar texturas 2D. Assim a imagem texturizada será colocada dentro de um modelo de garrafa 3D e todas as interações utilizadas para simular os efeitos dos movimentos manuais da ferramenta dos artesãos, que serão chamadas aqui de *pinceladas de areia*, serão feitas na interface da areia com a superfície interna da garrafa.

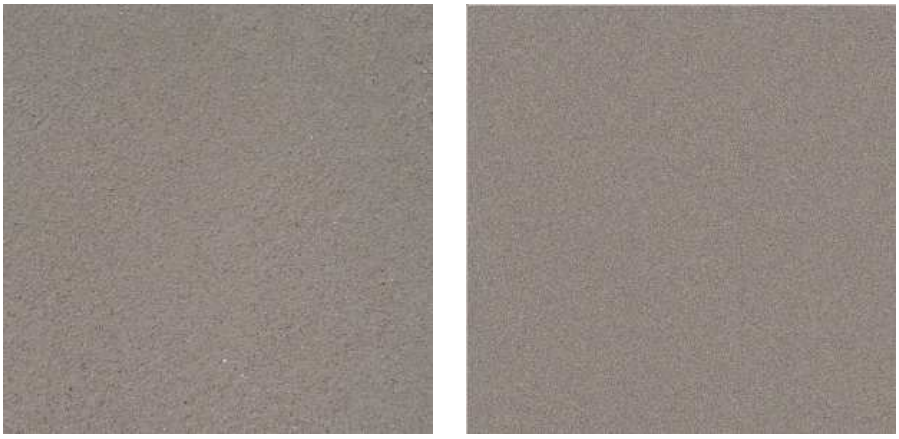
Agora será descrito o método usado para gerar as texturas de areia, que foi introduzido em Carvalho et al.[12]. Para gerar uma textura adequada para representar a areia artificial usada na geração dessas garrafas, foram fotografados vários exemplos de areia de várias cores usadas pelos artesãos. Notou-se, analisando essas imagens, que a variação do matiz é muito menor do que as variações observadas na saturação e no brilho e/ou intensidade. Desta forma, resolveu-se utilizar o sistema de cores HSV, decompondo as fotografias nos três canais de cores para analisar e determinar qual a melhor distribuição que as represente. Os histogramas dos exemplos de areia coletados, para todos os três canais, mostrou uma representação que se aproxima com uma precisão razoável de uma distribuição normal, também chamada de distribuição gaussiana.

Deste modo, a função de ruído gaussiano que foi utilizada para gerar perturbações nos valores dos pixels dos três canais do espaço de cores HSV, é dada por

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{4.1}$$

onde μ é a média e σ o desvio padrão. Considerou-se que os valores do HSV seriam

os valores das médias μ para cada canal, sendo que os valores dos canais do HSV, estão no intervalo de [0,255]. A conversão do intervalo dos canais HSV para o intervalo de [0,255] é transparente ao usuário ocorrendo internamente no programa. O usuário continua trabalhando com o canal H no intervalo de [0º,360º] e os canais S e V no intervalo de [0,255]. Gerou-se uma probabilidade cumulativa de cada canal para cada intensidade do intervalo. Quando os valores médios se aproximam dos extremos do intervalo, a função de probabilidade gera valores para intensidades fora do intervalo. Deste modo, move-se as probabilidades de intensidades de fora para dentro do intervalo, fazendo uma distribuição gaussiana truncada. Com isso, são gerados ruídos para cada um dos componentes HSV. Recompondo os canais, é produzida uma textura procedural de areia colorida. A Figura 4.3 mostra um exemplo de fotografia real de areia em (a) e a textura procedural de areia que simula a fotografia real em (b). Neste exemplo, pode-se observar como a textura sintética consegue simular de forma bem realística a fotografia real de areia.



(a) (b)

Figura 4.3: Fotografia real de areia em (a) e textura procedural de areia em (b).

Com esse método pode-se gerar várias texturas com cores diferentes, variando o HSV e seus respectivos desvios padrões. Foi desenvolvido um software para realizar a seleção das cores primárias das texturas de areia (ver Figura 4.4). Através dele, gerou-se as texturas de areia colorida para cada 30° no espaço do matiz, comparando visualmente a textura gerada com as fotografias, e armazenando-as na memória com o mesmo tamanho das imagens ou vídeos de entrada. Dentro de cada faixa de 30°, foram variados os valores de V e S criando texturas com cores mais claras e mais escuras. Com isso, foram produzidas um total de 23 cores primárias de texturas de areias. As cores e valores selecionados são exibidos na Tabela 4.1.

ID	CORES	H	S	V	AMOSTRA
01	preto	0º	0	35	
02	cinza	0º	0	128	
03	branco	0º	0	245	
04	vermelho	0º	170	195	
05	laranja	30º	170	195	
06	amarelo escuro	56º	182	60	
07	amarelo	56º	144	215	
08	verde claro	113º	113	182	
09	verde escuro	122º	144	60	
10	verde	122º	82	164	
11	verde água	150º	80	200	
12	ciano escuro	180º	118	106	
13	ciano	180º	53	235	
14	azul escuro	215º	200	80	
15	azul claro	210º	64	235	
16	azul	220º	111	237	
17	roxo escuro	270º	200	80	
18	roxo	270º	50	226	
19	magenta	322º	80	228	
20	rosa	347º	30	255	
21	bege	30º	30	235	
22	marrom	30º	50	100	
23	marrom claro	30º	50	130	

Tabela 4.1: Tabela de cores primárias das texturas de areia selecionadas. Os valores de desvio padrão $H\sigma$ = 3.1, $S\sigma$ = 5.3 e $V\sigma$ = 20.4 foram utilizados para todas as cores selecionadas. Os valores das médias são considerados como os valores dos canais HSV.

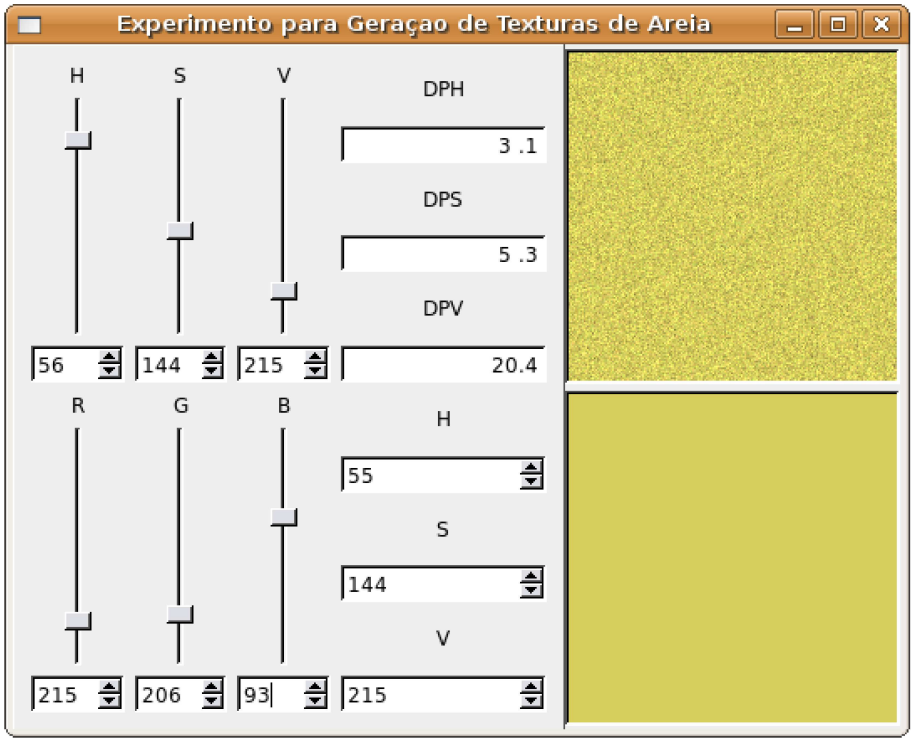


Figura 4.4: Ferramenta para seleção das cores primárias das texturas de areia.

4.2 Renderização Estilizada de Areia

Quando uma imagem é renderizada, computa-se, para cada pixel, o pixel correspondente em uma das texturas primárias de areia previamente armazenadas na memória, e desenha-se o valor do pixel da textura na imagem original na posição correspondente do pixel. Entretanto, há alguns casos especiais que devem ser tratados diferentemente, e para isso foram criados alguns limiares. Por exemplo, se o canal valor (V do modelo de cores HSV) do pixel é igual a 0, a cor é preta não importando os valores de H e S. Então criou-se um limiar, chamado de *limiar de valor baixo* ϕ_{lv}, para os pixels que tem o V muito próximo a 0, aplicando-se a textura preta de areia nos pixels com o V menor que este limiar, já que os pixels pertencentes a esta faixa tem as cores muito próximas a preto.

Outro limiar (*limiar de saturação baixa* ϕ_{ls}) foi criado para configurar o canal saturação, pois quando o S é igual a 0 o valor de H é indefinido, logo a cor do pixel só dependerá do valor de V, pertencendo à escala de tons de cinza. Então, quando o pixel tem S menor que o limiar de saturação baixa será aplicada uma textura de areia na escala de tons de cinza. Para a escala de tons de cinza foram criadas três texturas de areia: preta, cinza e branca. Aplica-se a textura cujo valor de V é mais próximo, ou cria-se uma mistura entre duas texturas que definam um intervalo contendo o valor de V. O usuário pode controlar a ocorrência ou não de misturas entre as texturas configurando limiares que serão explicados a seguir.

Se a cor do pixel está localizada entre duas cores primárias no espaço do matiz (H), calcula-se uma mistura entre duas texturas primárias correspondentes as duas cores primárias. Para isso, sorteia-se um número aleatório que determinará de qual textura de areia será o valor do pixel retornado. Por exemplo, se a cor do pixel sob análise pode ser decomposta em 30% de laranja e 70% de amarelo, e um número aleatório menor do que 0.7 (no intervalo de [0,1]) é sorteado para renderizar o pixel, o pixel terá o seu valor retornado da textura amarela, enquanto que se um valor maior que 0.7 for sorteado o pixel terá o seu valor retornado da textura laranja.

Para controlar as misturas foram criados dois limiares chamados de *limiar de mistura inferior* ϕ_{lm} e *limiar de mistura superior* ϕ_{um}. Esses limiares controlam, dentro de cada espaço entre duas cores primárias do matiz, o tamanho do intervalo onde ocorrerá misturas entre as texturas que representam essas duas cores. É necessário o controle das misturas para simular o processo manual de misturar areias com mais realismo, pois o artesão não faria misturas de um punhado de areia A com uma pitada de areia B. O artesão faz misturas em quantidades mais proporcionais. Por exemplo, se forem configurados os valores dos limiares de mistura inferior e superior em $\phi_{lm} = 30\%$ e $\phi_{um} = 60\%$, e o valor do canal H do pixel que está sendo processado, poder ser representado como uma mistura de 60% de uma textura A e 40% de uma textura B, então será sorteado um número aleatório para designar uma das duas texturas para representar o pixel, pois o valor do canal H do pixel, em relação a textura B, está entre o intervalo de [30%,60%]. Entretanto, se a quantidade de cor primária necessária para representar a cor do pixel na mistura está abaixo de 30% para a textura B, o valor do pixel será somente da textura A, ou se a cor do pixel na mistura está acima de 60% para a textura B, então a cor do pixel será somente da textura B. Pode-se ter essa visão em relação à textura A como limiares complementares, onde, quando o valor do matiz está abaixo de 40% (complementar de 60%) em relação à textura A, o valor renderizado será da textura B, e quando o valor for acima de 70% (complementar de 30%), o valor renderizado será da textura A. Dessa forma, um número aleatório não será sorteado para a escolha da textura, não ocorrendo a mistura. Quando os valores dos limiares de mistura são iguais ou o limiar inferior for maior que o superior, não haverá um intervalo para ocorrência de misturas no espaço do matiz.

A Figura 4.5 mostra um exemplo de mistura no espaço do matiz que fica entre as cores primárias laranja e amarelo, que correspondem às texturas A e B do exemplo anterior. A primeira linha mostra a cor original do matiz, e a segunda mostra a textura gerada por aquela cor, onde em (a) é formada por 100% da textura A e 0% da textura B, em (b) é formada por 83,33% da textura A e 16,67% de B, (c) 66,66% de A e 33,34% de B, (d) 49, 99% de A e 50,01% de B, (e) 33,34% de A e 66,66% de B, (f) 16,67% de A e 83,33% de B, e (g) 0% de A e 100% de B. Na segunda linha, o limiar de mistura inferior e superior estavam respectivamente com $\phi_{lm} = 0\%$ e $\phi_{um} = 100\%$. Na terceira linha foram geradas as texturas novamente, mas com o limiar inferior configurado para $\phi_{lm} = 30\%$ e

o superior em $\phi_{um} = 60\%$. Note que as texturas formadas abaixo de 30% para a textura B, foram totalmente renderizadas com textura A (laranja), que neste caso são (a) e (b). Note também que as texturas formadas acima de 60% para a textura B, que são (e), (f) e (g), foram totalmente renderizadas com a textura B (amarela). Se os limiares fossem $\phi_{lm} = 50\%$ e $\phi_{um} = 50\%$ para o superior não haveriam misturas e as texturas (a), (b) e (c) seriam laranjas e (d), (e), (f) e (g) amarelas.

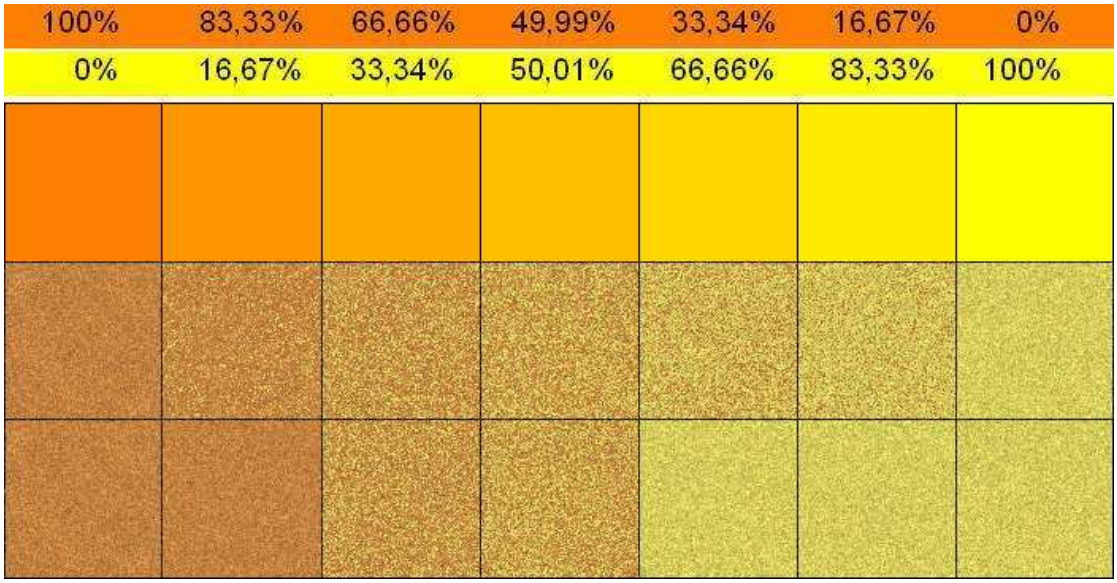


Figura 4.5: Exemplo de misturas entre texturas de areia. 1ª linha - cor original do matiz, 2ª linha - renderização com limiares de mistura $\phi_{lm} = 0\%$ e $\phi_{um} = 100\%$, e 3ª linha - renderização com limiares de mistura $\phi_{lm} = 30\%$ e $\phi_{um} = 60\%$.

Um problema de incoerência temporal aparece quando mistura-se duas ou mais texturas. Devido às texturas misturadas serem criadas em tempo de execução, o sorteio de um número aleatório para a sua criação, faz com que os valores dos pixels variem entre os quadros do vídeo, ou seja, a cada quadro é criada uma nova textura misturada que ocasiona texturas do mesmo tipo, mas com a valores diferentes para as posições de seus pixels. Ao assistir o vídeo nota-se um efeito indesejável de *flickering*. Este problema é resolvido em parte armazenando-se dinamicamente na memória a textura misturada e reutilizando-a para os quadros seguintes.

Como foi descrito anteriormente, o espaço do matiz foi dividido em faixas com ângulos de 30º. Para cada 30º, criou-se uma textura de areia colorida para a cor correspondente, além de algumas variações dessa cor. Estas variações da cor correspondente são obtidas pela variação dos canais S e V, que resultam em cores mais claras ou mais escuras. Desta forma, foram geradas um total de 23 cores primárias de texturas. Para controlar essa variação dos canais S e V, que informa se uma cor é mais clara ou mais escura, foram criados mais dois limiares, um para o S e outro para o V, que foram chamados de *limiar de saturação alta* ϕ_{hs} e *limiar de valor alto* ϕ_{hv}.

Os limires de saturação alta e valor alto são usados quando não é aplicada uma textura preta ou uma textura na escala tons de cinza. Deste modo, quando o valor do canal V do pixel em análise é menor que o limiar de valor alto será aplicada uma textura com a cor mais escura, devido estar mais próximo da textura preta, e quando o valor do canal V do pixel é maior que o limiar de valor alto, será aplicada uma textura com a cor mais clara, devido estar mais distante da textura preta. Da mesma forma ocorre com o limiar de saturação alta, quando o valor do canal S do pixel é menor que o limiar de saturação alta será aplicada uma textura com uma pigmentação mais fraca, e quando o valor do canal S do pixel for maior será aplicada uma textura com uma pigmentação mais forte, ou seja, mais pura. Com isso, podemos ter quatro tipos de texturas: com cor escura e pigmentação fraca, com cor escura e pigmentação forte, com cor clara e pigmentação fraca, e com cor clara e pigmentação forte. Logo, o pixel retornado da textura de areia irá depender da combinação entre os limiares de S e V (altos e baixos), além dos limiares de mistura (inferior e superior). Todos esses limiares podem ser configurados com a interação do usuário e seus resultados podem ser visualizados em tempo real com auxílio de uma interface gráfica. A Tabela 4.2 faz um resumo das funções de todos os limiares usados na renderização da areia.

Limiar	Variável	Função
valor baixo	ϕ_{lv}	determina quando será aplicada uma textura preta.
valor alto	ϕ_{hv}	se não for aplicada uma textura preta ou uma textura na escala de tons de cinza, determina quando será aplicada uma textura com uma cor mais clara ou mais escura.
saturação baixa	ϕ_{ls}	determina quando será aplicada uma textura na escala de tons de cinza (textura branca, cinza, preta ou uma mistura entre elas).
saturação alta	ϕ_{hs}	se não for aplicada uma textura preta ou uma textura na escala de tons de cinza, determina quando será aplicada uma textura com uma pigmentação mais fraca ou uma pigmentação mais forte.
mistura inferior	ϕ_{lm}	determina um limite inferior no espaço do matiz entre duas cores primárias, onde poderá ocorrer misturas dentro do intervalo determinado.
mistura superior	ϕ_{um}	determina um limite superior no espaço do matiz entre duas cores primárias, onde poderá ocorrer misturas dentro do intervalo determinado.

Tabela 4.2: Tabela com o resumo das funções dos limiares utilizados na renderização da areia.

Antes da renderização completa do vídeo, é necessário selecionar um de seus quadros para configurar os parâmetros que serão utilizados em todos os outros, tendo um *preview* de como ficará o resultado. O usuário pode então escolher o método de renderização que

achar mais adequado. Foram criados dois métodos de renderização, que são: *renderização por pixel* e *renderização por objeto*. Todos os valores pseudo-randômicos usados no processo de renderização das texturas são gerados anteriormente e armazenados na memória de textura, assim o processo inteiro pode ser feito através de *pixel shader* (instruções de softwares usadas para programação de unidades de processamento gráfico, que realizam operações a nível de pixel).

No método de renderização por pixel, para cada pixel da imagem de entrada, analisa-se o valor nos canais HSV a fim de escolher a textura mais apropriada para representá-lo. Após escolhida a textura, acessa-se a posição do pixel correspondente na textura de areia previamente armazenada na memória, e escreve-se o valor do pixel da textura na posição correspondente do pixel na imagem. Já no método de renderização por objeto, é necessária a segmentação dos objetos do vídeo. Nas segmentações usadas nos experimentos mostrados aqui, foi usado o algoritmo rápido de segmentação fuzzy [14], que é baseado nos trabalhos de Herman [37] e Carvalho et al. [13]. Para cada objeto do vídeo são usadas uma ou mais texturas de areia de acordo com a média calculada dos canais HSV dos pixels pertencentes ao objeto. Ao calcular a média do canal H, deve-se tomar cuidado com a cor vermelha, pois como H é um ângulo no intervalo de [0º, 360º], objetos vermelhos podem ter pixels próximos de 0º e próximos de 360º. Devido a este problema, a média dos objetos vermelhos serão em torno de 180º, ocasionando na escolha de uma textura mais próxima ao ciano do que ao vermelho. Para resolver este problema, quando for detectado que um objeto é vermelho, soma-se 360º aos pixels com valores próximos a zero e calcula-se a média de todos. Se o resultado da média for maior que 360º, seu valor é diminuido em 360º.

A renderização por objeto depende muito da segmentação do vídeo. Quando a segmentação não tem uma boa qualidade, a renderização não terá uma boa qualidade. Por exemplo, ao segmentar um objeto que possui pixels com cores muito diferentes entre si, ou se a segmentação selecionar pixels para o objeto que pertencem a objetos diferentes, no cálculo da média, a textura selecionada para renderização poderá não ser próxima a cor do objeto segmentado.Tanto na renderização por pixel quanto na renderização por objeto, pode-se utilizar a segmentação para selecionar parâmetros diferentes para cada objeto, além de tornar possível a aplicação de diferentes métodos de renderização para cada objeto, apesar dessa opção não ser usada nesse trabalho.

4.3 Simulando Efeitos de Pinceladas de Areia

Após derramar areia na garrafa e delinear alguns objetos, o artesão pode criar alguns efeitos na imagem através de "pinceladas" produzidas pela ferramenta. É importante simular esses efeitos na estilização do vídeo, se a intenção do usuário é produzir algumas imagens ou animações que são bem similares aos tipos de imagens vistas em garrafas

com areia colorida originais. Foram criados dois efeitos notáveis nas garrafas com areia colorida, introduzidos inicialmente por Britto Neto e Carvalho em [10], que foram denominados de: *efeito de altura e profundidade* e *efeito de vegetação*.

O efeito de altura e profundidade foi denominado assim pois é usado para expressar uma sensação vaga da variação da altura e/ou profundidade na composição da imagem. Um exemplo desse efeito pode ser visto na Figura 4.6. Para criar tal efeito, o artesão traz areia de diferentes cores do centro da garrafa para a superfície interna da garrafa, usando a ponta chata da ferramenta (veja Figura 4.2(a)), trazendo mais ou menos areia de acordo com o ângulo que a ferramenta faz com o eixo vertical. Definiu-se como 0^o quando a ferramenta traz mais areia, ou seja, quando uma das faces da ponta chata da ferramenta dentro da garrafa de vidro está com a largura máxima, e 90^o quando a ponta chata da ferramenta dentro da garrafa de vidro está com a largura mínima.

Figura 4.6: Exemplo do efeito de altura e profundidade. As curvas desenhadas pelo artesão usando diferentes tipos de areia dando uma sensação vaga da variação da altura e profundidade no objeto.

Para simular este efeito na produção das imagens pelo método de renderização estilizada proposta aqui, foram incluídos controles na ferramenta gráfica desenvolvida para mudar o ângulo de aproximação da ponta chata da ferramenta enquanto raspa o interior da parede do vidro. (Este ângulo refere-se ao ângulo de rotação da ferramenta em torno do eixo vertical.) Além de escolher o ângulo de aproximação, o usuário pode escolher a cor da areia que será arrastada para frente da areia original e indicar onde será aplicado o efeito. Dessa forma, ferramentas básicas de desenho foram adicionadas à ferramenta gráfica.

O usuário então desenha uma máscara indicando onde os efeitos serão aplicados, seleciona a textura de areia que será usada no efeito e o ângulo de aproximação da ferramenta. Um mapa de distância da máscara desenhada é calculado e seus valores são armazenados. Este mapa de distância é então usado para determinar se a areia selecionada substituirá a areia atual naquela posição ou não, de acordo com uma distribuição gaussiana com média 0 e desvio padrão dado pela equação

$$\sigma = \begin{cases} \sigma_{min}, & \text{se } 1 - \frac{\alpha}{\alpha_{max}} < t, \\ \sigma_{max}\left(1 - \frac{\alpha}{\alpha_{max}}\right), & \text{caso contrário,} \end{cases} \tag{4.2}$$

onde α e α_{max} são o ângulo que a ferramenta faz com a superfície interna da garrafa e o ângulo máximo permitido pela ferramenta (90º), respectivamente, t é um limiar configurado para limitar o valor do menor desvio padrão, e σ_{max} e σ_{min} são o máximo e o mínimo valores de desvio padrão permitidos, respectivamente. O valor do σ_{max} para a máscara foi configurado como o maior valor do mapa de distância da máscara. Então, baseando-se no valor do mapa de distância de um pixel, a probabilidade da distribuição gaussiana é calculada e um número aleatório é sorteado para determinar se o pixel será ou não substituído por um pixel da textura de areia selecionada ou se permanecerá com o valor do pixel da textura de areia associada antes da aplicação do efeito. Um exemplo de aplicação deste efeito pode ser visto na Figura 4.7.

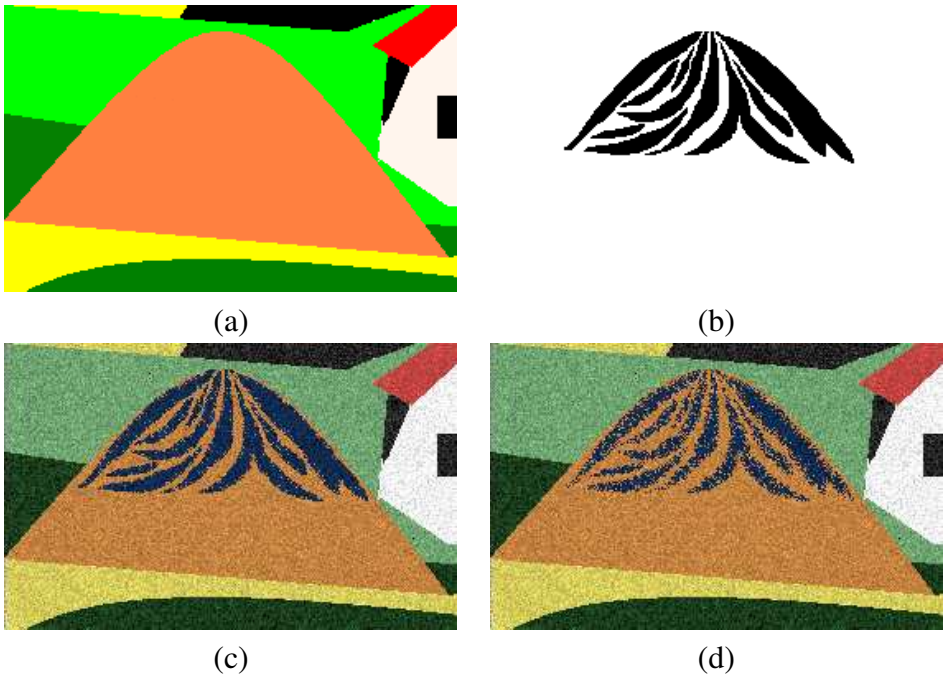
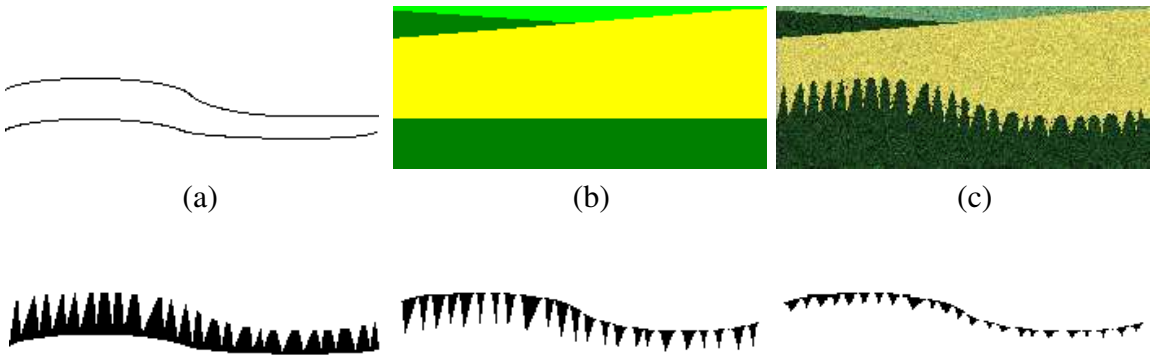


Figura 4.7: Exemplo da aplicação da simulação do efeito de altura e profundidade, onde são exibidos: a imagem original (a), máscara (b), efeito simulado com $\alpha = 20°$ (c) e $\alpha = 40°$ (d).

O segundo efeito criado pelos artesãos e simulado aqui foi o efeito de vegetação. Ele consiste em desenhar características de alta freqüência que representam uma vegetação na imagem. Os artesãos utilizam a ponta aguda da ferramenta para criar esse efeito. Um exemplo deste efeito pode ser visto na Figura 4.8.

Figura 4.8: Exemplo do efeito de vegetação (objeto verde).

Para simular este efeito, o usuário desenha, com ajuda da interface gráfica, duas linhas horizontais para delinear a variação da altura da vegetação simulada (veja Figura 4.9 (a)), e especifica a largura mínima e máxima da vegetação que será renderizada. O algoritmo então percorre a imagem verticalmente procurando pela primeira coluna onde ambas as linhas aparecem. A partir daí, ele vai desenhando quadriláteros entre as duas linhas, sendo que a largura do topo e da base são definidos por números aleatórios no intervalo de [1,min] para o topo e no intervalo de [min,max] para a base, onde min é a largura mínima em pixels e max a largura máxima em pixels que a base pode assumir, de acordo com uma distribuição uniforme, até que a última coluna forme um quadrilátero completo. O topo de cada quadrilátero é substituído pelo recorte da linha superior, fazendo com que o topo fique curvo ou retilíneo dependendo da linha desenhada. O quadrilátero se transforma em um triângulo quando a largura mínima for 1, pois o topo terá apenas 1 ponto na linha superior. Com isso, foi gerada uma máscara (Figura 4.9 (d)) que é invertida para gerar sua máscara complementar (Figura 4.9 (e)) formada apenas por triângulos.

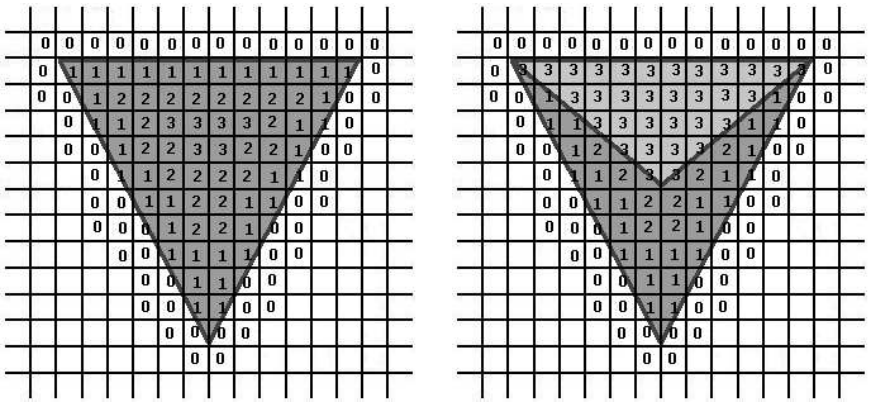


(a) (b) (c)

(d) (e) (f)

Figura 4.9: Linhas horizontais (a), imagem original (b), resultado (c), máscara gerada (d), máscara complementar (e) e máscara auxiliar (f).

Depois das máscaras geradas, é computado um mapa de distância modificado para a máscara complementar, de maneira similar ao usado no efeito de altura e profundidade, mas com uma pequena diferença. Para cada triângulo, depois que o mapa de distância padrão é calculado, calcula-se o centro de cada um, conectando cada centro com os vértices das respectivas bases dos triângulos, criando uma máscara auxiliar. Os valores do mapa de distância para essa máscara auxiliar são substituídos pelo valor máximo do mapa de distância (ver Figura 4.10). Isto é feito para limitar o efeito da mistura nas bordas verticais dos triângulos, deixando a área da base dos triângulos intacta. A renderização deste efeito é realizada para ambas as máscaras, sendo que a inferior é renderizada por completo e a superior é renderizada como no efeito de altura e profundidade, permitindo um pouco de mistura nas bordas verticais.



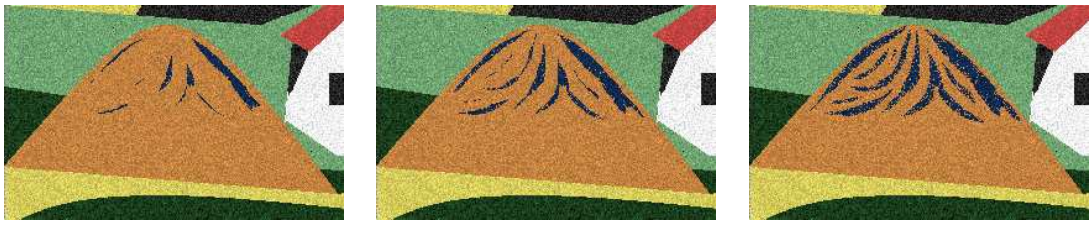
(a) (b)

Figura 4.10: mapa de distância (a) e mapa de distância modificado (b).

Quando estes efeitos são renderizados em vídeos, pode-se provocar problemas de incoerência temporal devido à propriedade aleatória desses efeitos. Por exemplo, tanto no efeito de altura e profundidade quanto no efeito de vegetação, nota-se um efeito indesejável de *flickering* devido à natureza aleatória das técnicas desenvolvidas. Dessa forma, ao renderizar o primeiro quadro do vídeo, armazena-se as máscaras geradas para a renderização dos próximos quadros, sem a necessidade de executar novamente o algoritmo por completo, forçando uma coerência temporal ao vídeo.

Se a intenção do usuário é criar efeitos estáticos, estas técnicas são adequadas. Entretanto, como o objetivo pode ser a criação de animações com esses efeitos, foi implementado um mecanismo simples que permite ao usuário incluir animações nos mesmos. O usuário define os quadros inicial e final do vídeo onde a seqüência irá aparecer, seleciona uma função (linear ou sigmóide) que tornará o efeito ligado ou desligado e um fator de inclinação (que dirá quão rápido o efeito irá aparecer ou desaparecer). Baseado nessa função e no fator de inclinação escolhido, os valores armazenados no mapa de distância (mapa de distância modificado no caso da vegetação, que poderá ser aplicado também na

máscara inferior), serão multiplicados por um fator (entre 0 e 1) antes da renderização do efeito ser realizada. A Figura 4.11 mostra 3 quadros de uma animação criada usando este método.



(a) (b) (c)

Figura 4.11: Três quadros da animação de um efeito.

4.4 Módulo de Renderização *Csand*

O software desenvolvido neste trabalho é um dos módulos de renderizações não fotorealísticas em vídeo de uma ferramenta maior denominada *AVP Rendering*. O *AVP Rendering* é uma ferramenta que permite ao usuário realizar segmentações em vídeo, cujo módulo de segmentação foi desenvolvido por Oliveira em [57], e a aplicação de novas técnicas ou técnicas existentes de renderizações não fotorealísticas. Neste trabalho, foi desenvolvido um módulo de renderização denominado *Csand*, do inglês *colored sand*, onde foi implementado uma nova metodologia para renderizações não fotorealísticas em vídeo que trabalha com areia colorida.

O módulo principal do *AVP Rendering* passa como entrada a imagem ou vídeo original para o *Csand*. A partir daí, o usuário pode renderizar a imagem ou vídeo por pixel e/ou por objeto. Na renderização por objeto, é necessário que o *Csand* receba como entrada o mapa de segmentação da imagem ou vídeo. Na renderização por pixel, pode-se também usar o mapa de segmentação, para que o usuário renderize apenas alguns objetos selecionados da imagem ou vídeo, e não todos os objetos, podendo deixar os objetos restantes para a aplicação de técnicas diferentes de renderização. Antes da renderização, o usuário pode aplicar efeitos de altura e profundidade, efeitos de vegetação e animar esses efeitos criados. Para isto, há um estágio de pré-renderização, onde as máscaras dos efeitos são aplicadas antes da renderização, permitindo que o usuário reutilize a pré-renderização para inserir novos efeitos. Após a renderização, a imagem ou o vídeo estilizado pode retornar a entrada do programa como a imagem ou vídeo original para a aplicação de novas renderizações de acordo com as intenções do usuário.

A Figura 4.12, mostra o *pipeline* de renderização do módulo *Csand*, que recebe como entrada uma imagem ou vídeo. As caixas tracejadas representam passos opcionais do processo (o passo de animação de efeitos pode aparecer somente se algum efeito de areia

estiver presente), enquanto que as linhas tracejadas de entrada representam uma entrada opcional para o *pipeline*.

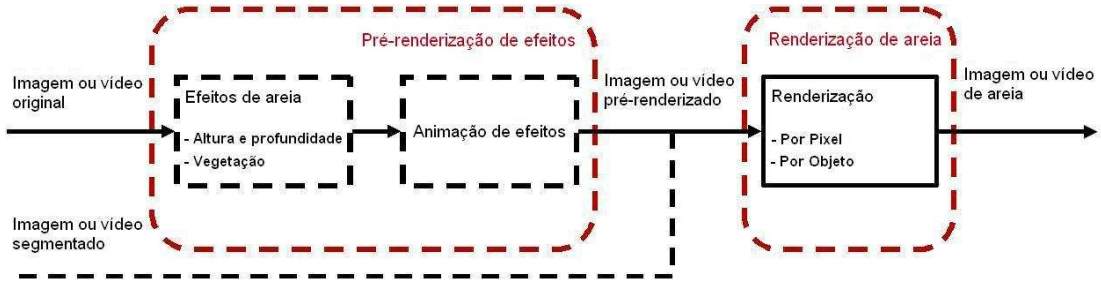


Figura 4.12: Pipeline de renderização *Csand*.

A seguir, através de um vídeo sintético, será mostrado o funcionamento do sistema e a estilização utilizando todos os efeitos desenvolvidos aqui. No final o vídeo será aplicado em um modelo de garrafa 3D. O vídeo criado para essa demonstração, é uma animação com imagens bem parecidas como as vistas em garrafas com areia colorida reais, onde um barco passeia da esquerda para direita no vídeo. A Figura 4.13 apresenta dois quadros do vídeo do barco. Nas imagens (a) e (b) estão os quadros originais, seguidos das imagens com o estágio de pré-renderização dos efeitos em (c) e (d) contendo a aplicação do efeito de altura e profundidade, a aplicação do efeito de vegetação em (e) e (f), e a aplicação de um efeito animado de altura e profundidade em (g) e (h) que neste caso está gerando as ondas do mar. Finalmente os quadros renderizados são mostrados em (i) e (j).

Para a aplicação do vídeo como textura em um modelo 3D de garrafa, é feito um *padding* (acolchoamento) no vídeo, na sua parte superior e inferior, com alguma textura de areia para o *background*, geralmente a cor com mais predominância no vídeo, para que o mesmo fique centralizado na parte cilíndrica da garrafa. A Figura 4.14 apresenta os dois quadros do barco dentro da garrafa, onde a garrafa foi rotacionada para acompanhar o movimento do barco.

Um problema da aplicação do vídeo na garrafa é o surgimento de *aliasing* [17] (a renderização de pequenos grãos de areia de forma quadriculada) que pode ocorrer se a câmera é posicionada muito próxima da garrafa . Este problema pode ser resolvido não permitindo que a câmera se aproxime muito da garrafa, mas está não é uma solução ideal. Uma outra maneira, mais apropriada para resolver o problema de *aliasing*, seria definir uma alta resolução na renderização da imagem de areia para a utilização de *mipmaps* [79] (a mesma imagem em várias resoluções) de acordo com a distância da câmera para a garrafa.

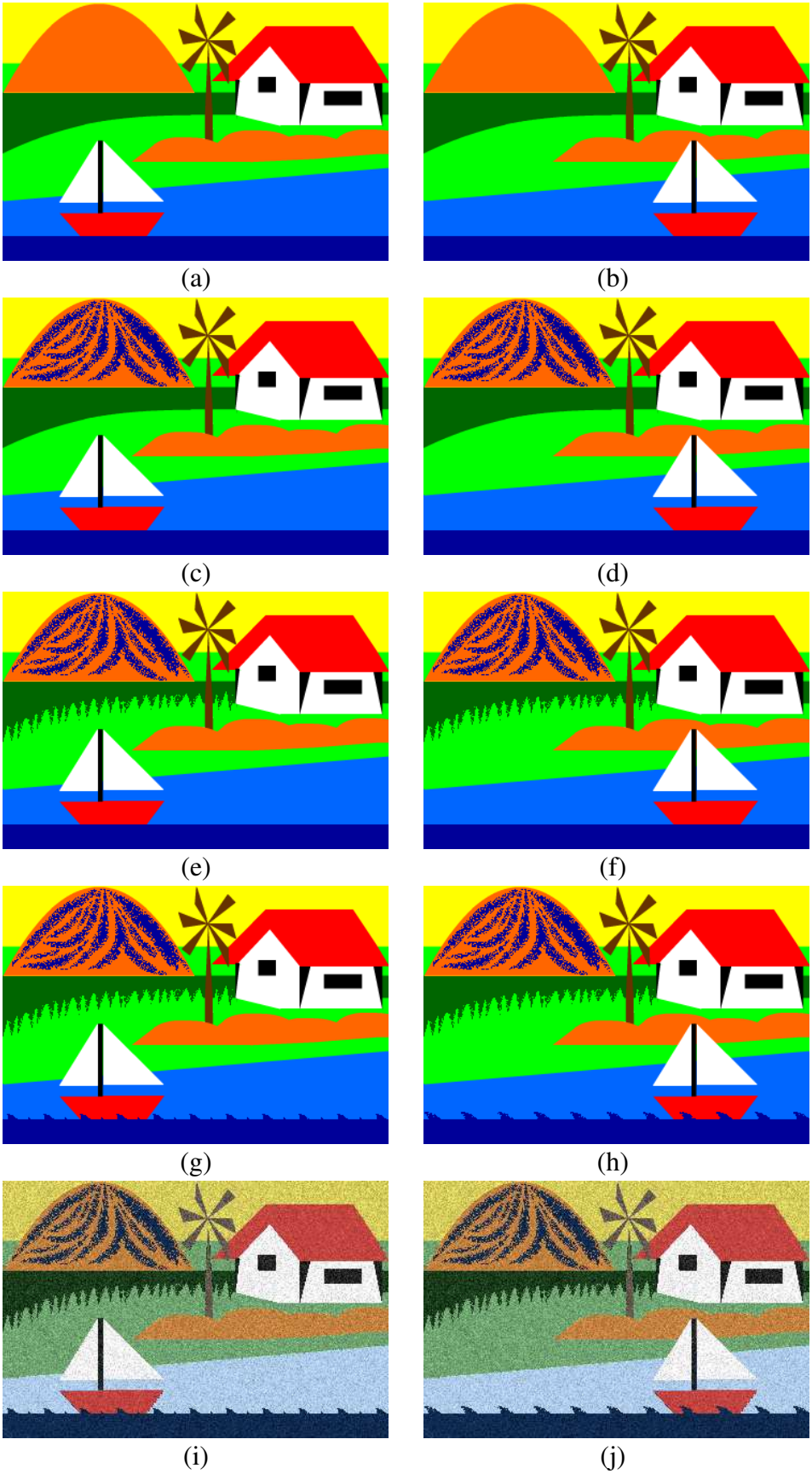


Figura 4.13: Dois quadros do vídeo do barco originais em (a) e (b), no estágio de pré-renderização em (c), (d), (e), (f), (g) e (h), e renderizados em (i) e (j).

(a)

(b)

Figura 4.14: Dois quadros do vídeo do barco dentro da garrafa.

4.5 Animações $2\frac{1}{2}$D em Caixas de Areia

Para a criação das animações $2\frac{1}{2}$D, adiciona-se ao video estilizado informações 3D através de um campo de alturas, onde o objeto que está em movimento causa deslocamentos nos grãos de areia de áreas estáticas, deslocando as colunas do campo de alturas verticalmente para que o objeto deixe seu rastro e crie amontoados de areia por onde passa. Na criação dessas animações utilizou-se uma abordagem semelhante com a de Sumner et al. [72].

Sumner et al. [72] introduzem um modelo para deformar materiais de terrenos como areia, lama e neve, onde objetos rígidos em 3D se movimentam sobre estes terrenos, deixando seus rastros ou pegadas, e dessa forma, criando pequenas animações na superfície do terreno. O modelo consiste em um campo de alturas, sendo as alturas definidas como colunas verticais de material de terreno. Usando algoritmos de deslocamento e compressão, são animadas as deformações que são criadas quando objetos rígidos fazem um impacto com o material do terreno deixando pegadas. Certas propriedades desse modelo podem ser variadas produzindo comportamentos diferentes para cada tipo de material de terreno. Neste modelo, Sumner et al. [72] descreve quatro etapas para a criação das animações, que são: detecção de colisões, deslocamento de materiais, erosão e geração de partículas. Na quarta etapa, Sumner et al. [72] utiliza um sistema de partículas para deslocar grãos de areia pelo ar, um efeito que não será utilizado nesta dissertação, por não ser interessante nas animações produzidas.

No método desenvolvido neste trabalho são necessárias seis etapas para criação das animações, a saber: segmentação do vídeo, estilização do vídeo com areia colorida, modelagem 3D do objeto de interesse, compressão, deslocamento e erosão.

A primeira etapa para gerar a animação é a extração do objeto de interesse no vídeo real que vai causar os deslocamentos na areia. Para isto utilizou-se o algoritmo rápido de segmentação *fuzzy* [14] para obtenção do mapa de segmentação. O mapa de segmentação possui as informações sobre quais áreas do vídeo o objeto está se movimentando, além de poder ser usado na estilização do vídeo com areia colorida.

A segunda etapa é a fase de estilização do vídeo, onde serão usada as técnicas desenvolvidas de renderização por pixel e/ou renderização por objeto, como foi explicado anteriormente, além de poder utilizar alguns efeitos de pinceladas de areia dependendo do tipo de animação que o usuário prentende gerar.

Para melhor visualização das etapas do método de geração das animações $2\frac{1}{2}$D será usado um exemplo prático. A Figura 4.15 mostra dois quadros do vídeo real em (a) e (b) que foram utilizados para exemplificar o processo, os mapas de segmentação *fuzzy* dos respectivos quadros são exibidos em (c) e (d), e os quadros do vídeo estilizado com areia colorida aparecem em (e) e (f). Neste vídeo, uma caneca vermelha se desloca da direita para esquerda do vídeo sendo puxada por um fio de náilon.

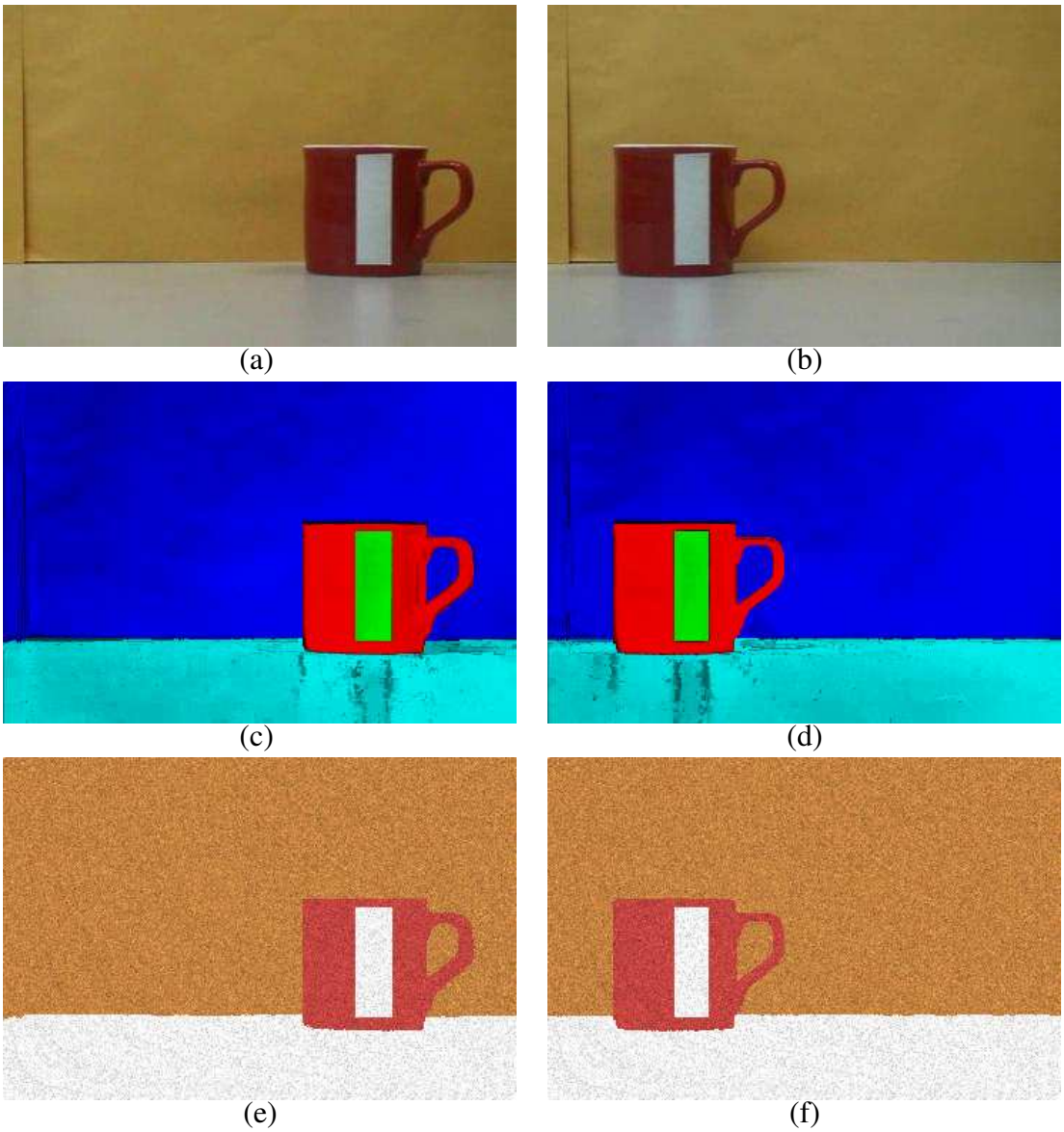
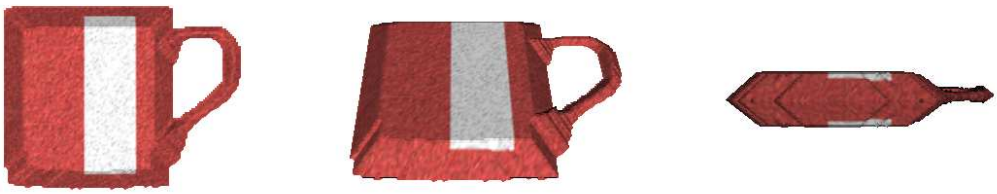


Figura 4.15: Dois quadros do vídeo da caneca em (a) e (b), os mapas de segmentação *fuzzy* em (c) e (d), e os quadros estilizados com areia colorida em (e) e (f).

É necessário definir um meio de modelar uma forma 3D para o objeto de interesse, mesmo que esta forma não seja a forma real do objeto. O objeto deve ter uma profundidade dentro da malha poliédrica, que representa o terreno, para que a malha sofra uma deformação, e também para calcular a quantidade de material que será comprimido e deslocado. Na criação do objeto 3D é calculado, a partir do mapa de segmentação, o mapa de distância do objeto, que é usado para definir uma profundidade ao objeto. O objeto 3D é formado por dois campos de alturas, um superior e outro inferior. Os valores das alturas, dos campos de alturas do objeto, são os valores do mapa de distância até uma profundidade definida. Se o mapa de distância tem valores maiores que a profundidade definida δ, os valores daquelas posições no campo de alturas serão iguais a δ. O

campo de altura inferior do objeto é formado por valores negativos e o campo superior por valores positivos. A Figura 4.16 mostra o objeto 3D extraído do vídeo da caneca, utilizando o método do mapa de distância para dar uma forma 3D abstrata, em algumas visões diferentes.



(a) (b) (c)

Figura 4.16: Forma 3D abstrata do objeto. Visão superior (a), visão em 45º (b) e visão de fundo (c).

Para cada pixel de um quadro do vídeo estilizado foi criado um vértice (coluna) da malha poliédrica com a cor do pixel correspondente, fazendo com que a malha tenha um número de colunas igual à altura multiplicada pela largura do vídeo.

A próxima etapa é a fase de compressão, onde as colunas que estão sob o objeto de interesse são comprimidas de acordo com a profundidade definida. Nesta fase, que corresponde a fase de detecção de colisões em Sumner et al. [72], o mapa de segmentação nos fornece as informações sobre quais colunas serão comprimidas e o mapa de distância nos fornece as informações sobre o quanto essas colunas serão comprimidas. Dessa forma, a simulação da superfície aproxima o movimento do material das colunas pela compressão ou deslocamento do material em baixo do objeto de interesse. Em cada quadro do vídeo, as alturas das colunas que estão na região do objeto de interesse e que ainda não sofreram compressão até a profundidade δ, são decrementadas até a superfície da malha ficar abaixo do objeto. Uma sinalização é configurada indicando que aquela coluna foi decrementada e o valor da mudança de altura é armazenado. Esse material decrementado é então comprimido ou forçado a se deslocar para uma coluna externa a área sob o objeto. Após isso, são feitas uma série de passos de erosões que são realizadas para reduzir a magnitude de inclinações entre as colunas vizinhas.

No deslocamento de materiais, o material é comprimido ou distribuído para as colunas ao redor do objeto. Uma taxa de compressão α, que é um valor controlado pelo usuário, é usada para calcular a quantidade de material que será distribuído, Δh, que é calculado por $\Delta h = \alpha m$, onde m é a quantidade total de material que foi deslocado. O material não comprimido é distribuído igualmente entre seus vizinhos com menor valor de distância (de acordo com o mapa de distâncias) até que todo o material fique depositado nas colunas ao redor do objeto, formando um anel de colunas. As colunas em volta do objeto têm suas alturas incrementadas de acordo com a quantidade de material depositado.

Na etapa de erosão, as colunas do anel são erodidas. O algoritmo de erosão identifica as colunas que formam inclinações excessivas com seus vizinhos e movem o material para

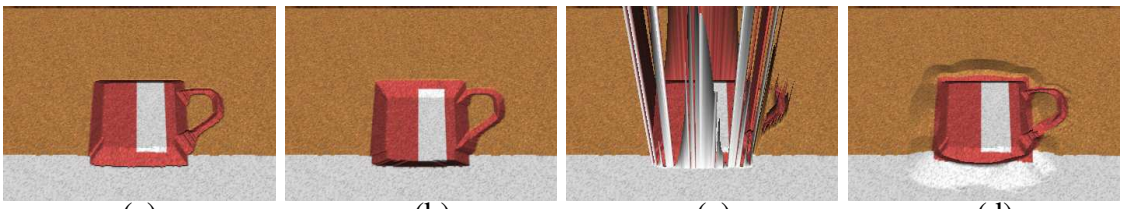
baixo da inclinação criando os amontoados de areia. Vários parâmetros são usados para controlar as formas dos amontoados e simular diferentes tipos de material de terreno. O algoritmo verifica a inclinação entre cada par de colunas adjacentes na malha. Para cada coluna ij e um vizinho kl, a inclinação s, é dada por

$$s = \tan^{-1}(h_{ij} - h_{kl})/d \tag{4.3}$$

onde h_{ij} é a altura da coluna ij e d é a distância entre as duas colunas. Se a altura entre as duas colunas vizinhas é maior que um limiar θ_{out}, então o material é movido da maior coluna para a coluna menor. Em um caso especial onde uma das colunas está em contato com o objeto, o objeto, um limiar diferente, θ_{in} é usado para fornecer um controle independente da inclinação interna. O material é movido usando-se o cálculo da diferença média das alturas, Δh_a, para as n colunas vizinhas com grande inclinação, que é dada por:

$$\Delta h_a = \frac{\Sigma(h_{ij} - h_{kl})}{n}. \tag{4.4}$$

A diferença média das alturas é multiplicada por uma constante fracionária, σ, e a quantidade resultante é igualmente distribuída entre os vizinhos. O algoritmo repete este passo até todas as inclinações ficarem abaixo de um limiar, θ_{stop}. A erosão pode provocar que algumas colunas cruzem o objeto, mas esta penetração é corrigida nos passos de tempo seguintes. A Figura 4.17 mostra as quatro etapas finais do processo para geração da animação $2\frac{1}{2}$D, na Tabela 4.3 observa-se os valores dos parâmetros utilizados na animação da caneca e, finalmente, a Figura 4.18 exibe três quadros da animação $2\frac{1}{2}$D do vídeo da caneca dentro de uma caixa de areia, sendo que os quadros (a) e (c) são os mesmo usados na Figura 4.15.



(a) (b) (c) (d)

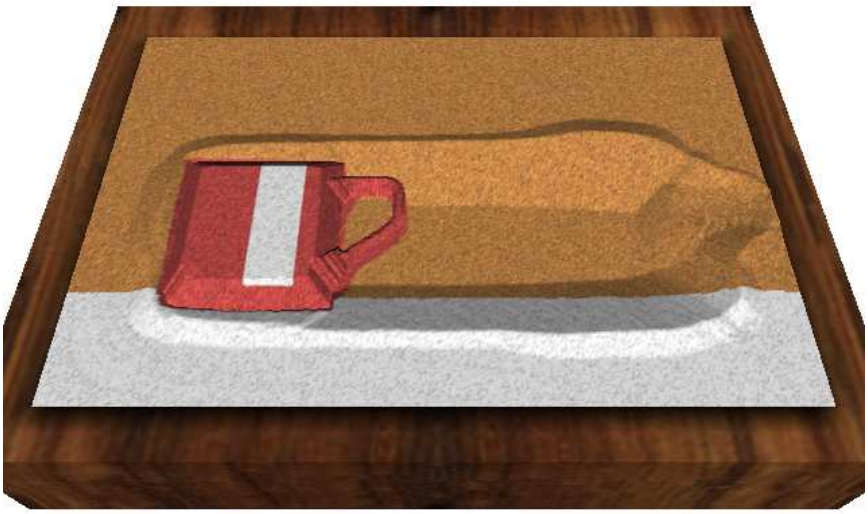
Figura 4.17: Modelagem 3D (a), compressão (b), deslocamento (c) e erosão (d).

Efeito	**Variável**	**Valor**
profundidade	δ	12
inclinação interna	θ_{in}	0.8
inclinação externa	θ_{out}	0.5
aspereza	σ	0.2
liquidez	θ_{stop}	0.8
compressão	α	0.3

Tabela 4.3: Tabela com valores dos parâmetros utilizados na animação da caneca.

(a)

(b)



(c)

Figura 4.18: Três quadros da animação $2\frac{1}{2}$D do vídeo da caneca dentro de uma caixa de areia.

As animações $2\frac{1}{2}$D em caixa de areia foram implementadas em OpenGL [55], mas podem ser facilmente implementadas em uma linguagem de programação de placas gráficas (*Graphics Processing Unit* ou GPU), como a linguagem Cg [56] (C for graphics) e a linguagem GLSL [61] (*OpenGL Shader Language*). A programação em GPU é muito utilizada em sistemas que realizam renderizações em tempo real [49].

Antes de iniciar a geração das animações $2\frac{1}{2}$D, o vídeo é segmentado por completo através do módulo de segmentação da ferramenta *AVP Rendering*. A partir de cada quadro do mapa de segmentação é obtido o mapa de distância do vídeo. O vídeo também é renderizado com areia colorida através do módulo de rederização *Csand*, onde cada quadro representará uma textura que será aplicada no campo de alturas. Os mapas de segmentação, os mapas de distância e os quadros estilizados do vídeo são passados como entrada para o programa que gera as animações. Desta forma, o programa, a cada quadro do vídeo, aplica a textura no campo de alturas, calcula as etapas de compressão, deslocamento e erosão, e exibe o resultado de cada quadro gerando a animação.

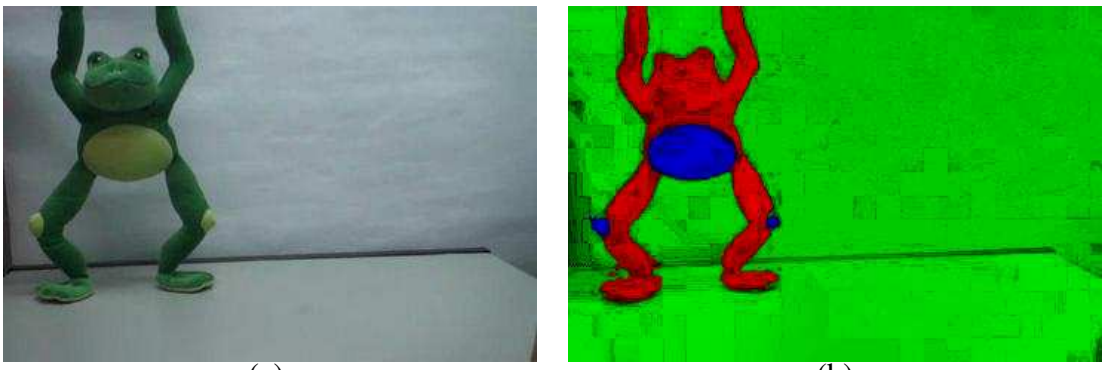
No início do processo para geração das animações $2\frac{1}{2}$D, o campo de alturas encontra-se totalmente plano. Quando uma coluna do campo de alturas tem a mesma altura do campo de alturas no estado inicial, define-se que a coluna tem uma altura ideal ou um nível ideal. Quando a coluna não tem uma altura ideal, ela sofreu uma deformação. Existem dois tipos de deformações no campo de alturas. O primeiro tipo é causada pela compressão do material abaixo do objeto de interesse, que foi denominada de depressão. O segundo tipo é causada pelo deslocamento de material seguido da erosão, que foi denominada de amontoação. As colunas que sofrem depressão ficam abaixo do nível ideal e as colunas que sofrem amontoação ficam acima do nível ideal.

Após o cálculo do primeiro quadro do vídeo, ocorre a primeira deformação no campo de alturas. No cálculo dos quadros seguintes, o objeto de interesse pode se movimentar em áreas que sofreram depressão e em áreas que sofreram amontoação. Nas áreas de amontoação, quando o processo está na etapa de compressão, deve-se armazenar o valor da mudança de altura com o valor da altura ideal, pois se for armazenado um valor maior do que a do nível ideal, ao calcular as etapas de deslocamento e erosão, ocorrerá a criação de amontoados desproporcionais à altura da primeira deformação, além de alterar o formato e a altura da deformação anterior, o que não é adequado para a animação. Nas áreas de depressão, as áreas que estão abaixo da profundidade δ não armazenam nenhum valor de mudança, e as áreas que estão abaixo da altura ideal e acima da profundidade δ, armazenam o valor de mudança de acordo com a diferença do valor atual da altura da coluna com o valor do mapa de distância. Isto é feito para que as áreas já comprimidas até a profundidade δ não sofram mais alterações.

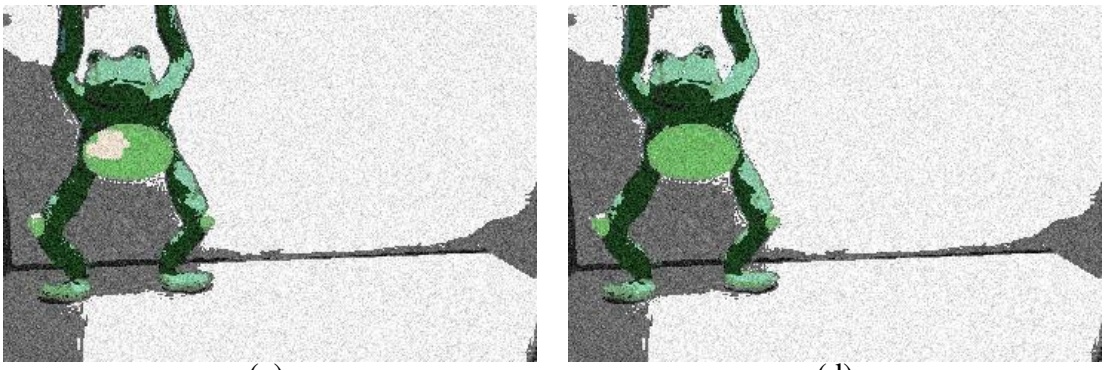
4.6 Experimentos

Agora é descrito alguns experimentos realizados para mostrar o funcionamento do método proposto. A Figura 4.19 exibe exemplos dos métodos de renderização por pixel e por objeto utilizados na estilização do vídeo do sapo já mencionado no Capítulo 3. A renderização por pixel é mais proveitosa quando deseja-se manter alguns detalhes da imagem. Neste exemplo, pode-se observar como a renderização por pixel mantém alguns pequenos detalhes dentro do corpo do sapo, tais como olhos, nariz, e parte da boca. Por outro lado, quando a intenção é limitar as cores usadas para renderizar um objeto, utiliza-se o método de renderização por objeto.

A Figura 4.19(b) mostra o mapa de segmentação do quadro original da imagem exibida na Figura 4.19(a), onde o matiz indica a qual objeto pertence o conjunto de pixels, e a intensidade (mapeada em cada objeto, no intervalo de [0,1]) reflete o grau de pertinência que o pixel tem em relação ao objeto. A Figura 4.19(c) mostra uma renderização por pixel do quadro, enquanto que a 4.19(d) foi gerada usando o mapa de segmentação mostrado em 4.19(b) para restringir as texturas usadas para renderizar os pixels pertencentes ao objeto azul (barriga e joelhos) para a textura verde. Para inserir detalhes na renderização por objeto, é necessário que objetos extras sejam adicionados no passo de segmentação. Por outro lado, a renderização por objetos pode ser usada para manter a aparência de todas as estruturas, forçando um certo nível de coerência temporal.



(a) (b)



(c) (d)

Figura 4.19: Um quadro do vídeo do sapo (a), o mapa de segmentação associado (b), a renderização por pixel do quadro (c), e o quadro onde o objeto azul de (b) foi renderizado usando o método de renderização por objeto (d).

Algumas imagens ou vídeos reais, após o processo de renderização estilizada, permanecem com muitos detalhes da imagem ou vídeo original. Alguns desses detalhes, muitas vezes, não são capturados pelo artesão no momento da composição das imagens na superfície interna da garrafa. Para deixar as imagens ou vídeos reais com o aspecto de que foram feitos manualmente, é necessário, às vezes, borrar as imagens ou quadros do vídeo real antes da renderização estilizada. Desta forma, cria-se uma versão abstrata da imagem ou vídeo real, eliminando muitos desses detalhes no momento da renderização.

Este processo é similar ao método proposto por Bousseau et al. em [8], que produz pinturas em aquarela a partir de fotografias. A Figura 4.20 apresenta uma imagem real em (a) que foi utilizada para exemplicar o processo de abstração. Em (b) é exibida a imagem abstrata. Na Figura 4.21, observa-se a renderização por pixel da imagem real em (a) e a renderização por pixel da imagem abstrata em (b) da Figura 4.20, onde percebe-se que a renderização da imagem abstrata perde um pouco a complexidade da imagem original, removendo alguns detalhes que certamente não serão reproduzidos pelos artesãos.

(a) (b)

Figura 4.20: Imagem original (a), e imagem abstrata (b)

A Figura 4.22 mostra dois quadros de uma animação dentro da garrafa, onde as imagens que compõem essa animação lembram o tipo de cena geralmente produzida pelos artesãos. Neste exemplo, utilizou-se um modelo 3D de garrafa similar às garrafas usadas pelos artesãos, mas outros modelos podem ser utilizados, desde que se tenha cuidado na sua escolha, pois alguns detalhes do modelo podem causar consideráveis distorções nas imagens renderizadas no cálculo da refração na renderização do vidro.

Por fim, as Figuras 4.23 e 4.24 mostram mais um exemplo de animação $2\frac{1}{2}$D em caixa de areia de um vídeo sintético, onde uma joaninha anda pela areia, deixando seu rastro formando a palavra "Aloha" que significa boas vindas no Havaí. A Figura 4.23 mostra os quadros originais do vídeo sintético em (a),(c) e (e), os respectivos quadros renderizados com areia em (b), (d) e (f), e dois quadros da segmentação utilizada em (g) e (h). A Figura 4.24 mostra os quadros da animação $2\frac{1}{2}$D. Na geração das animações $2\frac{1}{2}$D, deve-se tomar cuidado na escolha do vídeo, pois alguns vídeos não são interessantes para esse tipo de

animação. Um exemplo disso são vídeos onde o objeto de interesse está muito próximo da borda da caixa, o que causaria amontoados nessas áreas da borda, criando aberturas visíveis entre a caixa e o campo de alturas. O ideal é a escolha de vídeos onde os objetos de interesse são visto pela lateral e se movimentem de forma perpendicular à visão da câmera.

(a)

(b)

Figura 4.21: Imagem original renderizada com areia colorida (a), e imagem abstrata renderizada com areia colorida(b).

(a)

(b)

Figura 4.22: Dois quadros de uma animação que mostra o resultado da aplicação das técnicas de renderização com areia colorida dentro da garrafa.

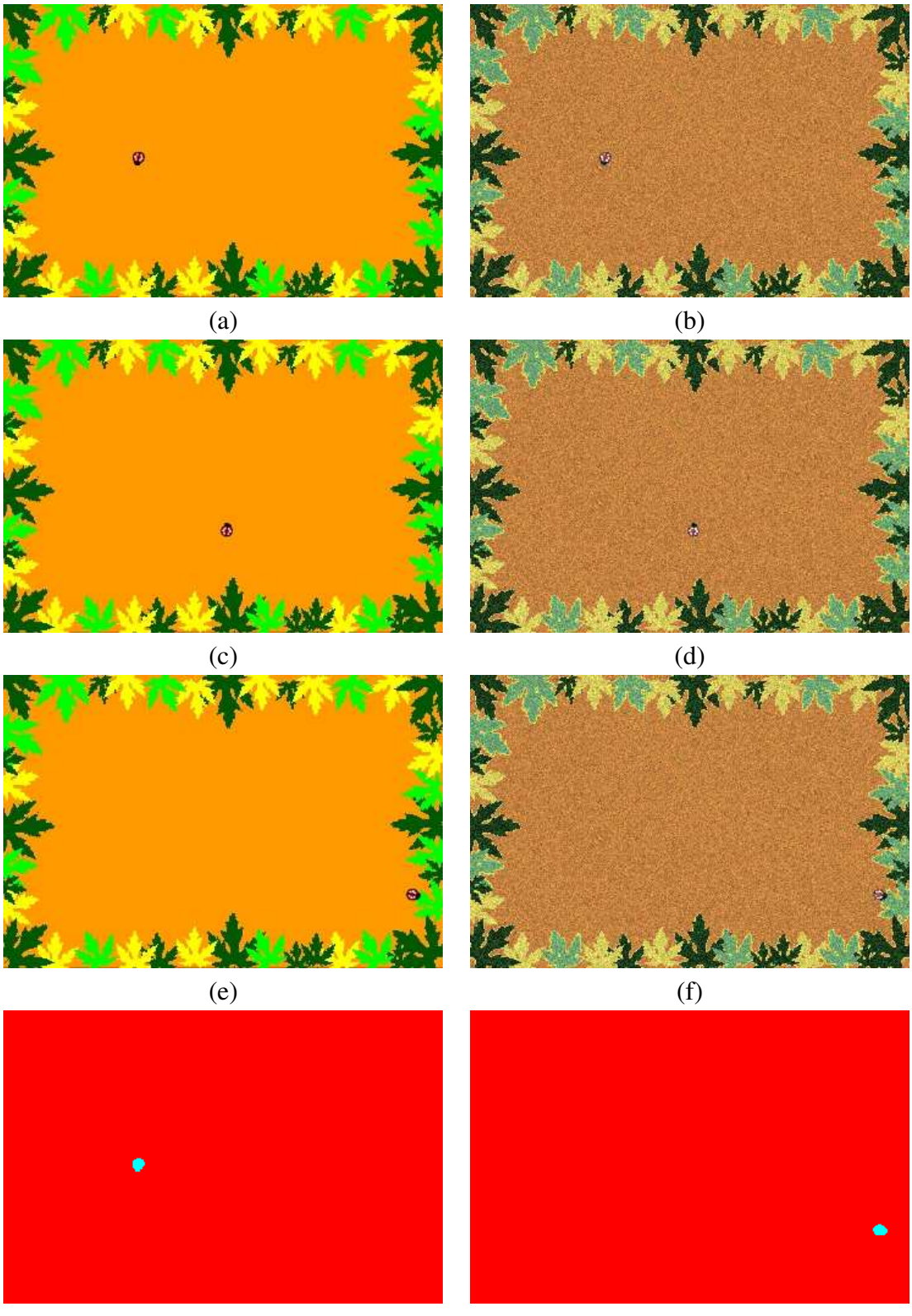


Figura 4.23: Três quadros do vídeo sintético da joaninha em (a), (c) e (e), os quadros renderizados em (b), (d) e (f), e dois quadros estilizados com os mapas de segmentação em (g) e (h).

(a)

(b)

(c)

Figura 4.24: Três quadros da animação $2\frac{1}{2}$D do vídeo sintético da joaninha.

Capítulo 5

Ferramenta *Csand*: Interface Gráfica do Usuário

Nesta dissertação foi desenvolvido um módulo de renderização não fotorealística denominado *Csand*, o qual recebeu o mesmo nome dado ao método desenvolvido. O módulo de renderização *Csand* faz parte de uma ferramenta maior denominada *AVP Rendering*, que trabalha com segmentação e renderizações não fotorealísticas para estilização de imagens e vídeos. Atualmente o *AVP Rendering* encontra-se em fase de integração de seus diversos módulos pelo grupo *AnimVideo*. O *AVP Rendering* é uma ferramenta que permite ao usuário realizar segmentações em vídeo e a aplicação de diversas técnicas existentes de renderizações não fotorealísticas. O módulo de segmentação do *AVP Rendering* foi desevolvido por Oliveira em [57].

A ferramenta foi implementada em linguagem C/C++ e sua interface desenvolvida em QT4 [7]. QT é um *framework* escrito em C++ para o desenvolvimento de aplicações com interface gráfica de usuário (*Graphical User Interface* ou GUI) que usa a metodologia *"write once, compile anywhere"* (escreva uma vez, compile em qualquer lugar), ou seja, ao escrever uma GUI com QT pode-se executar o software em qualquer plataforma como Windows 98 e XP, Mac OS X, Linux, Solaris, HP-UX, bastando apenas compilar novamente [7].

O módulo de renderização *Csand* é composto por duas abas. A primeira delas é a aba principal, onde o usuário interage com a ferramenta e, em tempo real, visualiza como a imagem ou quadro do vídeo vai ser estilizado após a renderização completa. A ferramenta possuí diversos *group boxes*, que são componentes de interface gráfica tipicamente usados como recipiente para outros controles de interface, agrupando os componentes com funções, atividades e/ou finalidades em comum através de bordas. Dessa forma, as imagens das telas capturadas (*screenshots*) da ferramenta foram numeradas pelos *group boxes* afim de tornar mais didática a apresentação da ferramenta. A Figura 5.1, mostra o *screenshots* da aba principal do módulo de renderização *Csand*.

A segunda aba foi denominada como aba de ferramentas. Nesta aba, estão embutidos

em seus componentes as técnicas utilizadas para simular os efeitos causados pelo uso da ferramenta dos artesões na composição das imagens dentro garrafas. A Figura 5.2, mostra o *screenshot* da aba de ferramentas do módulo de renderização *Csand*. Logo após cada *screenshot* da tela serão detalhados os respectivos componentes apresentados.

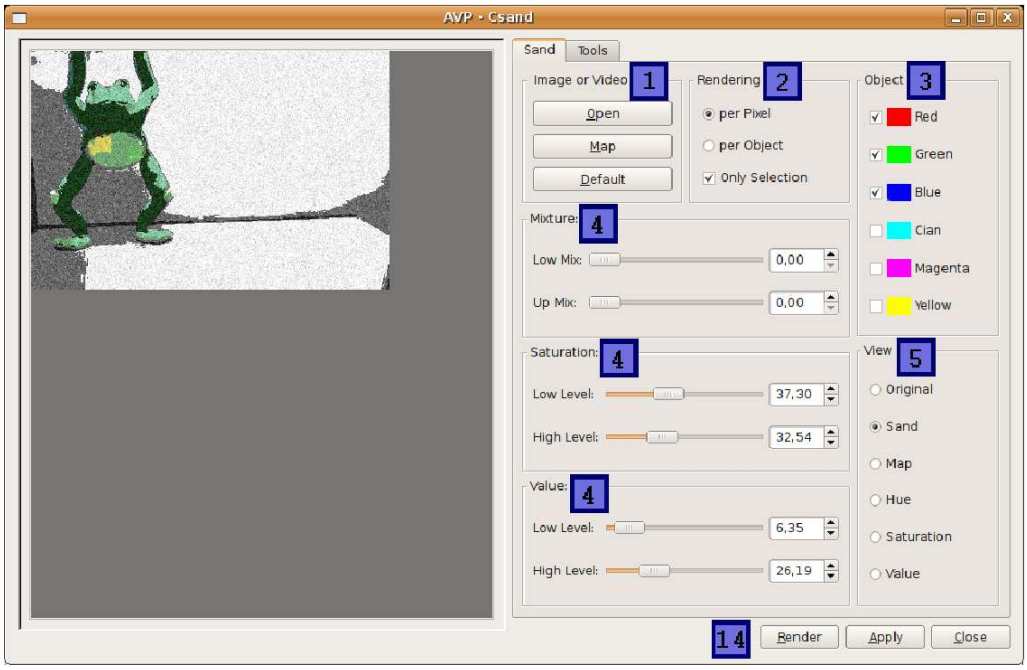


Figura 5.1: Tela da aba principal do módulo de renderização *Csand*

1. *Group Box - Image or Video*: Neste grupo se encontram os botões para carregar imagens ou vídeos, carregar o mapa de segmentação e um botão para reiniciar os valores dos limiares com os valores *defaults*, previamente definidos na implementação. Este grupo está presente devido o módulo ainda não estar totalmente integrado com o módulo principal da ferramenta *AVP Rendering*. Após o período de integração terminar, as imagens, vídeos e mapas de segmentação serão carregados apenas pelo módulo principal do *AVP Rendering*, que passará esses dados para o módulo de renderização *Csand* quando solicitado pelo usuário. O painel de visualização do módulo *Csand* mostra apenas o quadro do vídeo que foi escolhido para configuração dos limiares, funcionando como um *preview*. Os outros quadros são exibidos pelo painel de visualização do módulo principal do *AVP Rendering*.

2. *Group Box - Rendering*: Este grupo contém os componentes para a seleção de qual método de renderização será aplicado, que correspondem aos métodos de renderização por pixel ou renderização por objeto. Neste grupo encontra-se uma opção para permitir que o usuário selecione apenas alguns objetos do mapa de segmentação, onde, posteriormente, será aplicado o método de renderização escolhido. Com essa opção selecionada é possível misturar vídeo real com vídeo estilizado ou misturar diferentes estilos de renderizações não fotorealísticas prensentes no *AVP Rendering*.

3. *Group Box - Object*: Neste grupo é feita a seleção dos objetos da segmentação que serão renderizados. Futuramente, o *AVP Rendering* possuirá um módulo independente para a seleção de camadas e objetos que substituirá este grupo, podendo ser usado por todos os módulos de renderização.

4. *Group Boxes - Mixture, Saturation, Value*: Neste item da Figura 5.1 são marcados um conjunto com três *group boxes*, por terem funções similares, que correspondem aos seis limiares explicados durante o processo de renderização. A váriação desses limiares faz a seleção de quais texturas de areia colorida serão aplicadas na renderização da imagem. No painel de visualização do módulo *Csand* é exibido um *preview*, em tempo real, da renderização final da imagem.

5. *Group Boxes - View*: Neste grupo estão as opções do painel de visualização da aba principal, que pode ser alternado entre as visualizações do quadro original do vídeo, do quadro renderizado em areia, do mapa de segmentação do quadro, do matiz, da saturação e do valor do quadro escolhido.

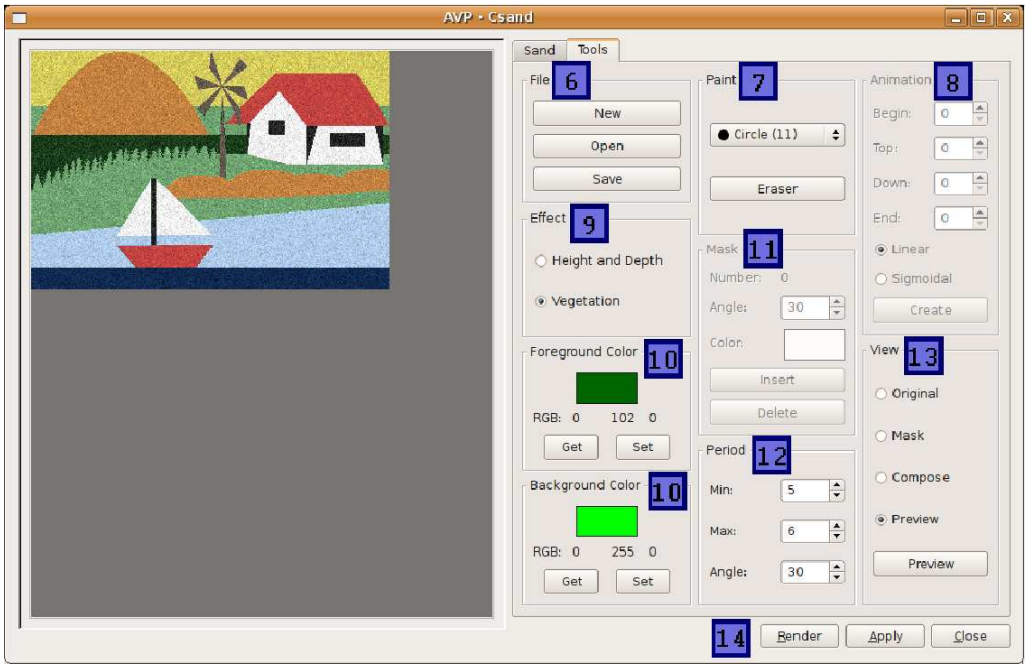


Figura 5.2: Tela da aba de ferramentas do módulo de renderização *Csand*.

6. *Group Box - File*: Aqui estão os botões usados para criar um novo efeito, carregar um arquivo contendo as configurações de efeitos armazenados e o botão para salvar um novo arquivo de configuração de efeitos. Os efeitos que podem ser criados e armazenados em arquivos de configuração são: o efeito de altura e pronfundidade, o efeito de vegetação e as configurações para a animação dos efeitos.

7. *Group Box - Paint*: Neste grupo, o usuário pode utilizar ferramentas básicas de pinturas para criação das máscaras dos efeitos. Uma ferrameta simples de *brush* é encontrada aqui com diversos tipos de tipos de *brush*, além de uma borracha.

8. *Group Box - Animation*: Neste grupo estão os controles para a animação dos efeitos criados. Através desses campos, o usuário informa os quadros inicial e final onde ocorrerá a animação, os quadros que funcionarão como pontos de inclinação em *top* e *down*, e a função que será aplicada para geração da animação.

9. *Group Box - Effect*: Neste grupo, o usuário escolhe o tipo de efeito, altura e profundidade ou vegetação, que será aplicado. Quando o usuário seleciona o efeito de vegetação a ferramenta de pintura passa a desenhar somente linhas com 1 pixel de largura.

10. *Group Boxes - Foreground Color, Background Color*: Aqui, o usuário tem a opção de escolher a cor da texura que será aplicada no *foreground* para os efeitos de altura e profundidade e vegetação, e de escolher a cor da textura de *background* para o efeito de vegetação.

11. *Group Box - Mask*: Neste grupo estão os cotroles para configuração de vários efeitos que ocorreram na imagem. Aqui, o usuário seleciona a cor que será usada na criação da máscara, e o ângulo utilizado pela ferramenta. Com isso, é possível criar várias máscaras, que aplicaram efeitos diferentes na mesma imagem, apenas inserindo uma nova cor. Por enquanto, este controles funcionam como uma pilha de máscaras, que podem ser adicionadas ou removidas, mas em versões futuras serão adicionados componentes para a edição e remoção de máscaras que se encontram em camadas inferiores da pilha. Estas informações sobre as máscaras podem ser salvas em arquivos de configurações.

12. *Group Box - Period*: Aqui é controlado o período do efeito de vegetação, onde são definidos as larguras mínima e máxima do efeito.

13. *Group Box - View*: Neste grupo estão as opções do painel de visualização da aba ferramentas, que pode ser alternado entre as visualizações do quadro original do vídeo, da máscara criada, da composição da máscara criada com o quadro original, e ter um *preview* de um dos quadros do vídeo renderizado em areia colorida com todos os efeitos aplicados.

13. *Botões - Render, Apply, Close*: E finalmente, são exibidos os botões com as seguintes funcionalidades: renderização do vídeo por completo, aplicação dos efeitos pré-configurados para geração das máscaras pré-renderizadas e a aplição de mais efeitos antes da renderização final, e fechar o módulo de renderização *Csand* retornando para o módulo principal do *AVP Rendering*.

Capítulo 6

Conclusão

Nesta dissertação propomos um método de renderização não fotorealística, que simula uma expressão artística típica da região Nordeste do Brasil, usando areia colorida para compor imagens de paisagens na superfície interna de garrafas de vidro. Uma técnica para geração de texturas procedurais de areia 2D foi descrita, e duas técnicas que imitam efeitos criados pelos artesãos na produção de garrafas preenchidas com areia colorida foram apresentadas. Propomos também um método para geração de animações $2\frac{1}{2}$D em caixas de areia, que utiliza uma abordagem parecida com a de Sumner et al. [72], onde os objetos que se movem no vídeo causam deslocamentos em grãos de areia de áreas estáticas do vídeo estilizado criando rastros e amontoados de areia por onde passam.

As técnicas descritas aqui são usadas para estilização de vídeos, algo quase impossível de ser produzido por um artesão na vida real. Com a estilização de vídeos com areia colorida é possível criar garrafas com areia colorida fotorealisticamente parecidas a uma garrafa com areia colorida real. Empregamos o algortimo rápido de segmentação *fuzzy* [14] para segmentar os vídeos como volumes 3D, e assim, permitir que o algoritmo de segmentação trate oclusão de objetos e objetos não temporalmente convexos. A coerência temporal nos vídeos estilizados são forçadas em objetos individuais através dos resultados da segmentação, pela restrição das texturas de areia usadas em alguns objetos.

As técnicas desenvolvidas foram usadas em vídeos reais e sintéticos, e vários quadros desses vídeos foram mostrados aqui, como imagens planas, sendo vistas dentro de uma caixa de areia, ou dentro de uma garrafa de areia, como apareceriam se fossem feitas por um artesão com habilidades neste estilo artístico. Estas técnicas são aplicadas na área de entretenimento, e podem ser utilizadas na produção de propagandas, vinhetas, filmes, animações para jogos e animações em geral.

Na estilização de vídeos com areia colorida foram encontrados alguns problemas de *flickering* [16] causados pela propriedade aleatória do cálculo, em tempo de execução, na geração das misturas entre texturas procedurais de areia e na criação das máscaras de pré-renderização. A solução para este problema de *flickering* [16] foi o armazenamento dinâmico das texturas misturadas e das máscaras de pré-renderização do quadro atual

para a aplicação nos quadros restantes. Outro problema encontrado foi de *aliasing* [17] , que ocorre quando a câmera é posicionada muito próxima da garrafa. A solução para o problema de *aliasing* [17] foi a utilização de uma alta resolução na renderização da imagem de areia para a utilização de *mipmaps* [79].

Implementamos um módulo de renderização não fotorealística denominado *Csand*, do inglês "*colored sand*". O *Csand* é uma ferramenta desenvolvida em linguagem C/C++, cujo sua interface foi desenvolvida em QT4 [7]. O módulo de renderização *Csand* faz parte de uma ferramenta maior denominada *AVP Rendering*, que trabalha com segmentação e renderizações não fotorealísticas para estilização de imagens e vídeos. Através do módulo de renderização *Csand* é possível transformar automaticamente vídeos reais capturados por uma câmera em animações feitas de areia colorida com uma pequena interação do usuário. Dessa forma, não é necessário que o usuário tenha conhecimentos de técnicas de animações para gerar uma animação de areia.

As animações $2\frac{1}{2}$D em caixa de areia foram implementadas em OpenGL [55], mas podem ser facilmente implementadas em uma linguagem de programação de placas gráficas (*Graphics Processing Unit* ou GPU), como a linguagem Cg [56] (C for graphics) e a linguagem GLSL [61] (*OpenGL Shader Language*). A programação em GPU é muito utilizada em sistemas que realizam renderizações em tempo real [49]. As técnicas de renderização estilizada com areia colorida também podem ser implementas em *pixel shader* (instruções de softwares usadas para programação de GPU, que realizam operações a nível de pixel).

Como trabalho futuro iremos rever a estrutura da micro-textura de areia e usar um *sistema de partículas* [60], onde cada partícula corresponderia a um grão de areia. Dessa forma, o problema de *aliasing* [17] seria resolvido. Por outro lado, o uso de um sistema de partículas aumentaria o tempo de processamento na computação dos milhares de grãos de areia, que puderam ser simulados de forma realista utilizando uma simples textura procedural de areia 2D.

Outra sugestão seria o uso de uma abordagem parecida com a utilizada por Snavely et al. [65] na geração das animações $2\frac{1}{2}$D em caixas de areia. Snavely et al. [65], combina os benefícios de entradas 2D e 3D pelo uso de diferentes tipos de entrada, como o vídeo com informações de profundidade em cada pixel, ou o vídeo $2\frac{1}{2}$D. Com a captura de vídeos $2\frac{1}{2}$D é potencialmente fácil a construção de complexos modelos 3D, o que substituiria a forma 3D abstrata sugerida nas animações $2\frac{1}{2}$D do método proposto nesta dissertação. Informações de profundidade de alta qualidade podem ser adquiridas de várias maneiras, como as chamadas ZCams [40], para a captura da profundidade, que estão disponíveis comercialmente, e sistemas que foram desenvolvidos para a captura de formas de alta resolução espacial e temporal, como o sistema desenvolvido por Zhang et al. [84].

Referências Bibliográficas

[1] APPEL, A., ROHLF, F. J., AND STEIN, A. J. The haloed line effect for hidden line elimination. In *SIGGRAPH '79: Proceedings of the 6th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1979), ACM Press, pp. 151–157.

[2] ARCELLI, C., AND DI BAJA, G. S. Ridge points in Euclidean distance maps. *Pattern Recogn. Lett. 13*, 4 (1992), 237–243.

[3] AZEVEDO, E., E CONCI, A. *Computação Gráfica: Teoria e Prática*, primeira ed. Elsevier, 2003.

[4] BEAUCHEMIN, S. S., AND BARRON, J. L. The computation of optical flow. *ACM Comput. Surv. 27*, 3 (1995), 433–466.

[5] BELL, N., YU, Y., AND MUCHA, P. J. Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), ACM Press, pp. 77–86.

[6] BEVILACQUA, A. Effective object segmentation in a traffic monitoring application. In *Third Indian Conference on Computer Vision, Graphics & Image Processing* (2002), pp. 125–130.

[7] BLANCHETTE, J., AND SUMMERFIELD, M. *C++ GUI Programming with Qt 4*, first ed. Prentice Hall, 2006.

[8] BOUSSEAU, A., KAPLAN, M., THOLLOT, J., AND SILLION, F. X. Interactive watercolor rendering with temporal coherence and abstraction. In *NPAR '06: Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2006), ACM Press, pp. 141–149.

[9] BOVIK, A. C., Ed. *Handbook of Image and Video Processing*, second ed. Academic Press, 2005.

[10] BRITTO NETO, L. S., AND CARVALHO, B. M. Message in a bottle: Stylized rendering of sand movies. In *SIBGRAPI '07: Proceedings of the 20th Brazilian*

Symposium on Computer Graphics and Image Processing (Los Alamitos, CA, USA, 2007), IEEE CS Press., pp. 11–18.

[11] BRUNET, P., AND SCOPIGNO, R., Eds. *Eurographics '99: Proceedings of the 20th Annual Conference ot the European Association of Computer Graphics*. Blackwell Publishers, Oxford, UK, 1999.

[12] CARVALHO, B. M., BRITTO NETO, L. S., AND OLIVEIRA, L. M. Bottled sand movies. In *CGIV '06: Proceedings of the International Conference on Computer Graphics, Imaging and Visualization* (Sidney, Australia, July 2006), IEEE CS Press, pp. 402–407.

[13] CARVALHO, B. M., HERMAN, G. T., AND KONG, T. Y. Simultaneous fuzzy segmentation of multiple objects. *Discrete Applied Mathematics 151*, 1-3 (2005), 55–77.

[14] CARVALHO, B. M., OLIVEIRA, L. M., AND SILVA, G. S. Fuzzy segmentation of color video shots. In *Proceedings of the Digital Geometry for Computer Imagery* (London, 2006), vol. 4245, Springer-Verlag, pp. 402–407.

[15] CHEN, C. T. Video compression: standards and applications. *Journal Communication and Image Sentation 4*, 2 (1993), 103–111.

[16] COLLOMOSSE, J. P., ROWNTREE, D., AND HALL, P. M. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics 11* (2005), 540–549.

[17] CROW, F. C. The aliasing problem in computer-generated shaded images. *Commun. ACM 20*, 11 (1977), 799–805.

[18] CUNNINGHAM, S., BRANSFORD, W., AND COHEN, M. F., Eds. *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 1998.

[19] CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. Computer-generated watercolor. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 421–430.

[20] DAVIES, E. R. *Machine Vision: Theory, Algorithms, Practicalities, 2nd Ed.* Academic Press, London, 1996.

[21] DI BLASI, G., AND GALLO, G. Artificial mosaics. *The Visual Computer 21* (2005), 373–383.

[22] DOBASHI, Y., HAGA, T., JOHAN, H., AND NISHITA, T. A method for creating mosaic images using Voronoi diagrams. In *Proceedings of Eurographics 2002 Short Presentations* (2002), Eurographics, pp. 341–348.

[23] EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. *Texture and Modeling: A Procedural Approach. 2nd Ed.* Morgan Kaufmann, San Francisco, 2003.

[24] FAUSTINO, G. M., AND DE FIGUEIREDO, L. H. Simple adaptive mosaic effects. In *SIBGRAPI '05: Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing* (Washington, DC, USA, 2005), IEEE Computer Society, p. 315.

[25] FEKETE, J.-D., AND SALESIN, D., Eds. *NPAR '00: Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering*. ACM Press, New York, NY, USA, 2000.

[26] FINKELSTEIN, A., AND SALESIN, D. H. Multiresolution curves. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM Press, pp. 261–268.

[27] FOLEY, J. D., DAM, A. V., FEINER, S. K., AND HUGHES, J. F. *Computer Graphics. Principle and Practice*, second ed. Addison-Wesley, 1990.

[28] GOMES, R., SANTOS, T. S., E CARVALHO, B. M. Coerência temporal intra-objeto para NPR utilizando fluxo ótico restrito. *Revista Eletrônica de Iniciação Científica*, no. II. junho 2007.

[29] GONZALEZ, R. C., AND WOODS, R. E. *Digital Image Processing*, second ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[30] GOOCH, A., GOOCH, B., SHIRLEY, P., AND COHEN, E. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM Press, pp. 447–452.

[31] GOOCH, B., AND GOOCH, A. *Non-Photorealistic Rendering*, first ed. A K Peters, 2001.

[32] GOOCH, B., SLOAN, P.-P. J., GOOCH, A., SHIRLEY, P., AND RIESENFELD, R. Interactive technical illustration. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM Press, pp. 31–38.

[33] GOURAUD, H. Continuous shading of curved surfaces. *IEEE Transactions on Computers* (1971), 623–629.

[34] HAEBERLI, P. Paint by numbers: abstract image representations. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), ACM Press, pp. 207–214.

[35] HANRAHAN, P., AND KRUEGER, W. Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM Press, pp. 165–174.

[36] HEARN, D., AND BAKER, M. P. *Computer Graphics with OpenGL*, third ed. Pearson Prentice Hall, 2004.

[37] HERMAN, G. T. Multiseeded segmentation using fuzzy connectedness. *IEEE Transactions on Pattern Analysis and Machine Inteligence 23* (2001), 460–474.

[38] HERTZMANN, A., AND PERLIN, K. Painterly rendering for video and interaction. In *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering* (2000), pp. 7–12.

[39] HORN, B. K. P., AND SCHUNCK, B. G. Determinig optical flow. *Artificial Intelligence 17* (1981), 185–203.

[40] IDDAN, G., AND YAHAV, G. 3D imaging in the studio (and elsewere...). In *Proceedings of SPIE '01* (2001), vol. 4298, pp. 48–55.

[41] JIN, S., LEWIS, R. R., AND WEST, D. A comparison of algorithms for vertex normal computation. *The Visual Computer 21*, 1-2 (Feb 2005), 71–82.

[42] JINDŘICH PARUS, A. H., AND KOLINGEROVÁ, I. Temporal face normal interpolation. In *SIGRAD '06. The Annual SIGRAD Conference, Special Theme: Computer Games* (Linköping, Sweden, 2006), Linköping University Electronic Press, pp. 12–16.

[43] KHAN, S., AND SHAH, M. Object based segmentation of video using color motion and spatial information. In *Computer Vision and Pattern Recognition* (2001), vol. 2, IEEE, pp. 746–751.

[44] LAKE, A., MARSHALL, C., HARRIS, M., AND BLACKSTEIN, M. Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00: Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2000), ACM Press, pp. 13–20.

[45] LANSDOWN, J., AND SCHOFIELD, S. Expressive rendering: A review of nonphotorealistic techniques. *IEEE Computer Graphics and Applications 15*, 3 (May 1995), 29–37.

[46] LEE, Y.-H., AND HORNG, S.-J. The chessboard distance transform and the medial axis transform are interchangeable. In *IPPS '96: Proceedings of the 10th International Parallel Processing Symposium* (Washington, DC, USA, 1996), IEEE Computer Society, pp. 424–428.

[47] LINKLATER, R. *Waking Life DVD*. Twentieth Century Fox Home Video, 2001.

[48] LITWINOWICZ, P. Processing images and video for an impressionist effect. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 407–414.

[49] MARK, W. R., GLANVILLE, R. S., AKELEY, K., AND KILGARD, M. J. Cg: a system for programming graphics hardware in a C-like language. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM Press, pp. 896–907.

[50] MATSUYAMA, T., AND PHILLIPS, T. Y. Digital realization of the labelled voronoi diagram and its application to closed boundary detection. In *Proc. of 7th International Conference on Pattern Recognition* (1984), pp. 478–480.

[51] MCCANE, B., NOVINS, K., CRANNITCH, D., AND GALVIN, B. On benchmarking optical flow. *Computer Vision and Image Understanding 84* (2001), 126–143.

[52] MCGUIRE, M., AND SIBLEY, P. A heightfield on an isometric grid. Tech. Rep. CS-05-14, Department of Computer Science at Brown University, Oct 2005.

[53] MEIER, B. Painterly rendering for animation. In *Proceedings of the ACM SIGGRAPH* (1996), pp. 477–484.

[54] MOULD, D. A stained glass image filter. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 20–25.

[55] NEIDER, J., DAVIS, T., AND WOO, M. *OpenGL Programming Guide*, second ed. Silicon Graphics, 1994.

[56] NVIDIA. *CG Toolkit User's Manual*, 1.4 ed., 2005.

[57] OLIVEIRA, L. M. Segmentação fuzzy de imagens e vídeos. Dissertação de Mestrado, Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Fevereiro 2007.

[58] PERLIN, K. Improving noise. *ACM Transactions on Graphics 21* (2002), 681–682.

[59] PROESMANS, M., GOOL, L. J. V., PAUWELS, E. J., AND OOSTERLINCK, A. Determination of optical flow and its discontinuities using non-linear diffusion. In *ECCV '94: Proceedings of the Third European Conference-Volume II on Computer Vision* (London, UK, 1994), Springer-Verlag, pp. 295–304.

[60] REEVES, W. T. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2 (1983), 91–108.

[61] ROST, R. J. *The OpenGL Shading Language*, second ed. Addison-Wesley Professional, 2006.

[62] SAITO, T., AND TAKAHASHI, T. Comprehensible rendering of 3-d shapes. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, TX, USA, 1990), ACM Press, pp. 197–206.

[63] SALISBURY, M. P., ANDERSON, S. E., BARZEL, R., AND SALESIN, D. H. Interactive pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM Press, pp. 101–108.

[64] SMITH, A. R. Color gamut transform pairs. In *SIGGRAPH '78: Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1978), ACM Press, pp. 12–19.

[65] SNAVELY, N., ZITNICK, C. L., KANG, S. B., AND COHEN, M. Stylizing 2.5-D video. In *NPAR '06: Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2006), ACM Press, pp. 63–69.

[66] SOUSA, M. C. *Computer-Generated Graphite Pencil Materials and Rendering*. PhD thesis, Department of Computing Science, University of Alberta, 1998.

[67] SOUSA, M. C., AND BUCHANAN, J. W. Observational model of blenders and erasers in computer-generated pencil rendering. In *Graphics Interface '99* (June 1999), I. S. MacKenzie and J. Stewart, Eds., pp. 157–166. ISBN 1-55860-632-7.

[68] SOUSA, M. C., AND BUCHANAN, J. W. Observational models of graphite pencil materials. *Computer Graphics Forum 19*, 1 (March 2000), 27–49. ISSN 1067-7055.

[69] SOUSA, M. C., AND BUCHANAN, J. W. Computer-generated graphite pencil rendering of 3d polygonal models. *Computer Graphics Forum 18*, 3 (September 1999), 195–208.

[70] STRASSMANN, S. Hairy brushes. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), ACM Press, pp. 225–232.

[71] STROTHOTTE, T., AND SCHLECHTWEG, S. *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*, first ed. Morgan Kaufmann, 2002.

[72] SUMNER, R., O'BRIEN, J. F., AND HODGINS, J. K. Animating sand, mud, and snow. *Computer Graphics Forum 18*, 1 (March 1999), 17 – 26.

[73] SUTHERLAND, I. E. Sketchpad: A man-machine graphical communication system. In *Proceedings of the 1963 Spring Joint Computer Conference* (1963), E. C. Johnson, Ed., vol. 23, American Federation of Information Processing Societies, Spartan Books, pp. 329–346.

[74] VILARIÑO, D. L., CABELLO, D., PARDO, X. M., AND BREA, V. M. Video segmentation for traffic monitoring tasks based on pixel-level snakes. In *1st Iberian Conference on Pattern Recognition and Image Analysis* (Puerto de Andratx, Espanha, 2003), vol. 2652, Lectures Notes in Computer Science, pp. 1074–1081.

[75] WANG, J., BHAT, P., COLBURN, R. A., AGRAWALA, M., AND COHEN, M. F. Interactive video cutout. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM Press, pp. 585–594.

[76] WANG, J., XU, Y., SHUM, H. Y., AND COHEN, M. F. Video tooning. *ACM Transactions on Graphics 23* (2004), 574–583.

[77] WARD, V. *What Dreams May Come DVD*. Polygram Filmed Entertainment, 1998.

[78] WATT, A., AND POLICARPO, F. *The computer image*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[79] WILLIAMS, L. Pyramidal parametrics. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1983), ACM Press, pp. 1–11.

[80] WINKENBACH, G., AND SALESIN, D. H. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM Press, pp. 91–100.

[81] WINKENBACH, G., AND SALESIN, D. H. Rendering parametric surfaces in pen and ink. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM Press, pp. 469–476.

[82] WINNEMÖLLER, H., OLSEN, S. C., AND GOOCH, B. Real-time video abstraction. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 1221–1226.

[83] YE, Q. Z. The signed Euclidean distance transform and its applications. In *Proceedings of the 9th International Conference on Pattern Recognition* (Rome, Italy, November 1988), vol. 1, IEEE Computer Society Press, pp. 495–499. ICPR'88.

[84] ZHANG, L., CURLESS, B., AND SEITZ, S. M. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 2003), pp. 367–374.

[85] ZHAO, H. Fast accurate normal calculation for heightfield lighting on a non-isometric grid. In *CGIV '06: Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 408–413.

[86] ZHU, Y., AND BRIDSON, R. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM Press, pp. 965–972.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo