3
Inferindo Geometrias 3D a partir de Imagens 2D

Em computação gráfica, o artista controla o processo de renderização mudando a iluminação, o material, a textura e o sombreamento. Os principais componentes para iluminar um ponto de uma superfície são a posição do ponto e a normal à superfície nesse ponto. Em desenhos feitos à mão, a normal da superfície é desconhecida e a informação de posição carece de profundidade [25].

Embora a assistência do computador na animação tradicional tenha ganhado muita atenção nos últimos anos, ela ainda possui uma série de limitações. Parte das pesquisas atuais focam no emprego de modelos tridimensionais, que são iluminados em diferentes estilos de sombreamento. As desvantagens dessa abordagem são as necessidades de criar modelos 3D e a grande dificuldade em reproduzir movimentos realistas.

Este capítulo descreve uma linha de produção (*pipeline*) para inferir geometrias tridimensionais a partir de imagens bidimensionais a fim de permitir-lhes a aplicação de técnicas de sombreamento evitando a real transformação do objeto para uma geometria 3D. O processo é especialmente desenvolvido para imagens criadas através do processo tradicional de animação (celulose) sendo proposto como maneira de minimizar a intervenção do usuário. A Figura 3.1 mostra os principais estágios dessa linha de produção.

3.1
Digitalização

Parte do sucesso da animação tradicional é devido ao seu aspecto carismático. Os traços, movimentos e formas não-realistas dos desenhos animados transportam o público para um mundo de fantasia. Este mundo de formas e movimentos não regidos pelas leis do mundo real é um dos principais atrativos dessa arte. Embora exista o desejo de transmitir um mundo imaginário, cada traço do desenho é extremamente preciso e importante. É

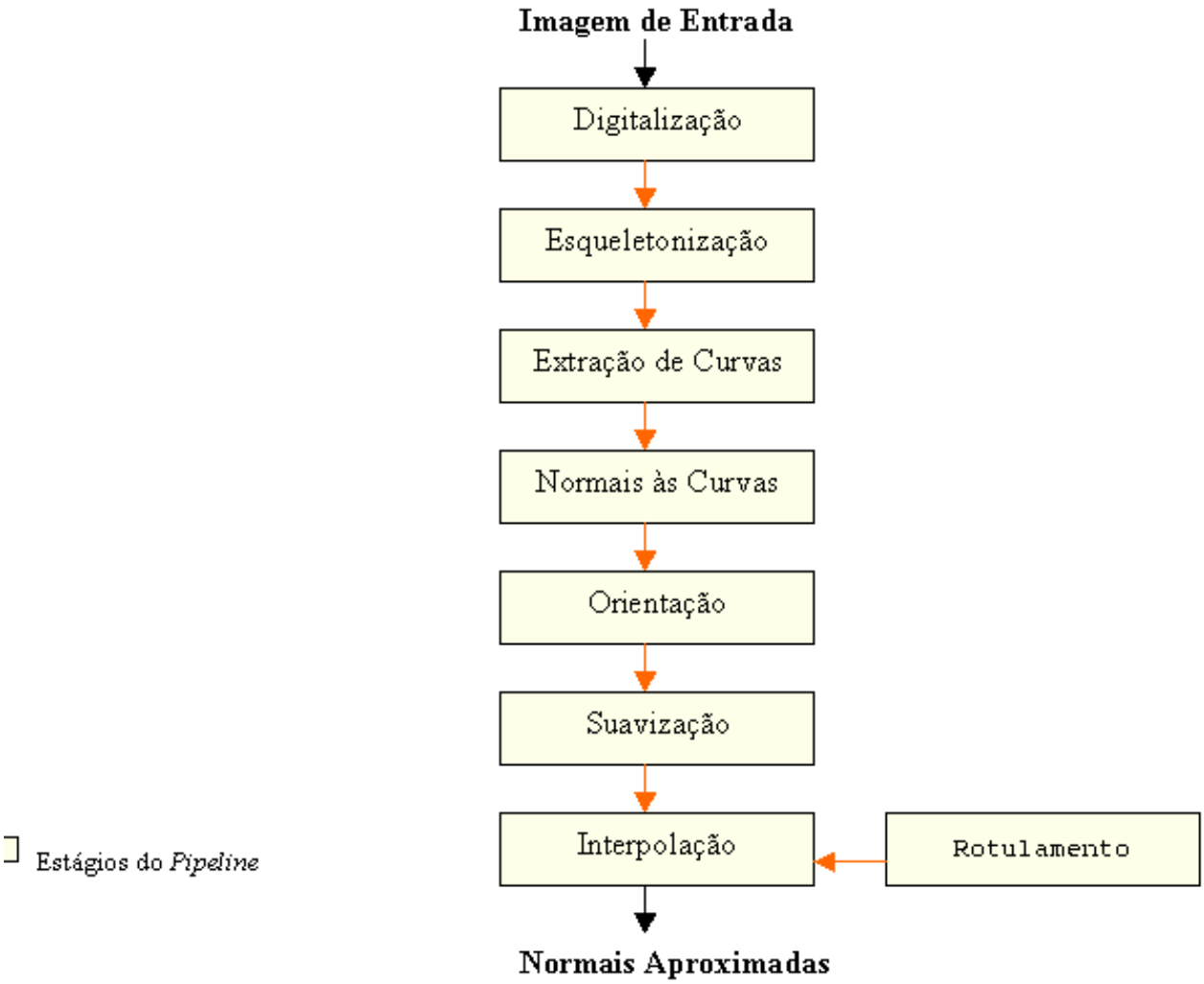

Figura 3.1: Linha de produção de pré-processamento de imagens 2D.

através das silhuetas que uma variedade de sentimentos é transmitida ao público.

No processo tradicional de animação, animadores começam desenhando com papel e lápis as seqüências de animação, uma imagem ou "quadro" por vez. Visando dar suporte a este modelo de produção, a linha de produção recebe como entrada desenhos originais de uma animação. As imagens são transferidas do papel para um meio digital através do uso de *scanners*. Considera-se que os desenhos originais são homogêneos, sem textura e desenhados sobre fundo branco com tinta suficientemente escura. Esses requisitos facilitam a posterior limpeza da imagem, onde, ao final do processo, devem possuir contorno necessariamente preto sobre fundo branco.

Após o processo de digitalização, cada célula possui furos de registro. Esses furos são pequenos buracos ao longo da margem superior ou inferior do papel que permitem a célula ser colocada em prendedores no *scanner*

garantindo o alinhamento entre as células. Se as células não estiverem bem alinhadas, a animação, quando tocada em alta velocidade, vai parecer "deslocada". A Figura 3.2 mostra um exemplo de entrada na linha de produção após o processo de limpeza.

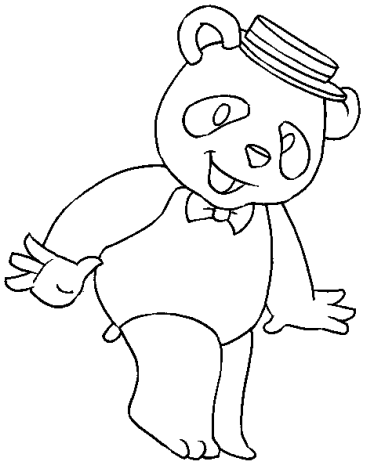

Figura 3.2: Entrada de dados da linha de produção.

3.2 Esqueletonização

A melhoria da qualidade e inteligibilidade de uma imagem é obtida através do uso de técnicas de processamento da imagem. Essas técnicas realçam determinadas características que são relevantes para o objetivo final do uso da imagem.

Em imagens que consistem principalmente de linhas, é a posição e orientação das linhas e o relacionamento entre elas que fornecem informações importantes sobre a imagem. *Pixels* sobressalentes são elementos que complicam a análise, sendo desejável, portanto, sua remoção. Para facilitar a extração das características da imagem, uma esqueletonização – um subconjunto dos *pixels* das linhas da imagem, espacialmente colocado ao longo do seu eixo medial, que contém suas propriedades topológicas e reproduz sua forma global – é computado. Assim, algoritmos de esqueletonização "corroem" a imagem reduzindo objetos sem mudar sua topologia. Por exemplo, considere os ursos ilustrados na Figura 3.3; ambos possuem a mesma forma, porém, a imagem da direita não possui *pixels* sobressalentes e pode ser melhor utilizada para análises baseadas em formas.

Existem vários algoritmos de esqueletonização disponíveis na literatura como, por exemplo: [26], [27]. Neste trabalho, o algoritmo de *Zhang-*

Suen, descrito em [28], foi utilizado por ser eficiente, de simples entendimento e possuir bom custo computacional.

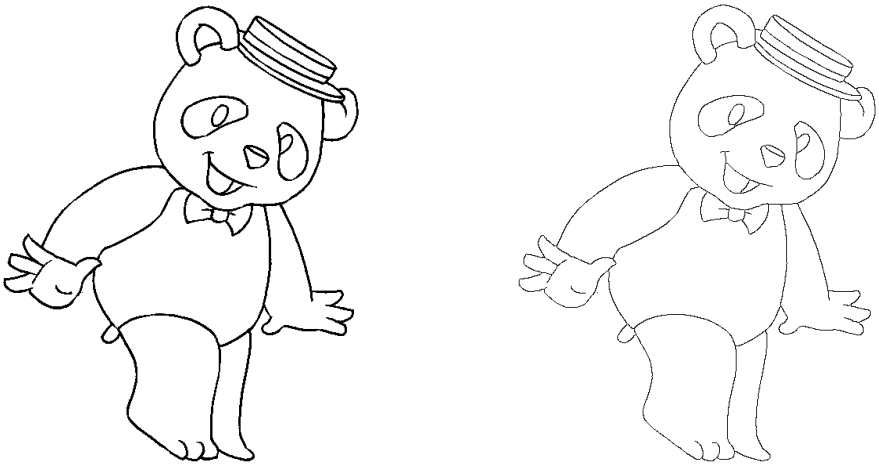

Figura 3.3: O processo de esqueletonização aplicado à imagem à esquerda remove *pixels* sobressalentes e produz imagens mais simples para análise futura.

3.3 Extração de Curvas

O contorno de um objeto é uma região onde a intensidade da imagem muda rapidamente. Ao detectar-se esta região, denominada aresta, consegue-se discernir os objetos de uma imagem. Após o processo de esqueletonização, uma imagem será bem representada pelo conjunto de curvas nela contido. Uma curva arbitrária é composta por uma seqüência de vetores unitários com um conjunto limitado de possíveis direções, como mostra a Figura 3.4. Para extrair informações sobre a estrutura morfológica dos objetos da cena, cada curva da imagem é codificada utilizando o algoritmo de *chain code* com grade 8-conectada descrito em [28].

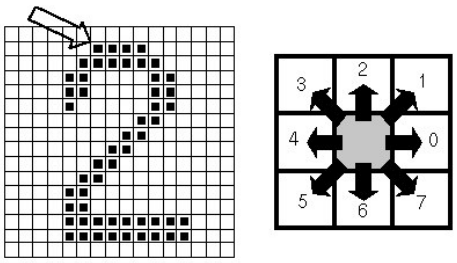

Figura 3.4: Um exemplo de *chain code*.

3.4 Normais às curvas

Uma vez identificadas as curvas da imagem, é necessário determinar os vetores normais em cada um de seus pontos. De posse da estrutura morfológica da curva (obtida através do algoritmo de *chain code*), o vetor normal em cada ponto é facilmente determinado tomando-se o vetor perpendicular ao vetor direção da curva. Os vetores normais podem estar em qualquer espaço, porém é conveniente transformá-los para o espaço da tela (X, Y) com profundidade Z. Para uma projeção ortográfica, a componente z das normais ao longo da aresta de uma superfície curva deve ser zero para manter a normal perpendicular ao vetor do olho.

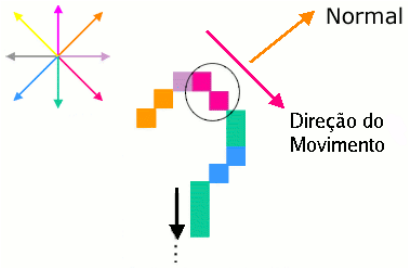

Figura 3.5: Identificação de vetores normais a uma curva.

3.5 Orientação

A orientação de uma curva é um fator importante na determinação dos vetores normais nos pontos desta. Um ponto em uma curva pode possuir dois vetores normais que partilham da mesma direção mas possuem sentidos opostos. A direção desse vetor determina a orientação da superfície. Como exemplo, se as normais na aresta de um círculo apontam para fora de sua curvatura, a interpolação desses vetores ao longo do interior do círculo gera um campo de vetores normais que define uma superfície convexa. Da mesma maneira, se for escolhida a direção que aponta para dentro da curvatura, uma superfície côncava será definida. Nesta dissertação, um método estima, para cada segmento, qual a melhor orientação analisando sua curvatura dominante e escolhendo aquela que aponta para fora da curva, como mostra a Figura 3.5.

3.6 Suavizacão

O algoritmo de *chain code* utilizado para representar as curvas da imagem utiliza uma vizinha 8-conectada como dito anteriormente. Dessa forma, é fácil ver que existem apenas 8 direções possíveis de vetores normais em cada ponto da curva. Faz-se necessário, então, uma leve "perturbação" na direção desses vetores a fim de obter uma suavização. Este estágio torna suave o campo de normais da seguinte forma: para cada ponto, o vetor normal é calculado pela adição dos vetores normais de seus pontos vizinhos anterior e posterior. A Figura 3.6b) ilustra os vetores normais, exibidos no espaço RGB, estimados para a imagem exibida em a). A transformação para o espaço RGB mapeia as componentes X, Y, Z em R, G, B respectivamente.

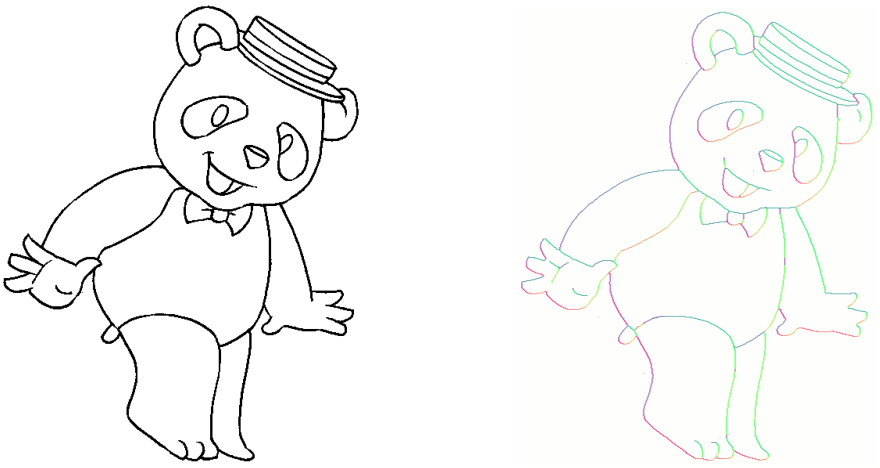

Figura 3.6: Estimando vetores normais. (a) imagem original. (b) vetores normais no espaço RGB.

3.7 Rotulamento

O rotulamento de uma imagem corresponde à identificação de regiões na imagem não-sobrepostas, cada uma sendo significante de acordo com uma aplicação em particular. Nesta seção é apresentada uma técnica de rotulamento que particiona uma imagem através de uma algoritmo de preenchimento.

Dado um ponto numa imagem, um algoritmo de *flood fill* identifica (recursivamente) todos os *pixels* adjacentes a este ponto que possuem atributos (cor) semelhantes. Quando aplicado a partir de um ponto no interior de uma curva fechada, esse algoritmo termina por identificar todos

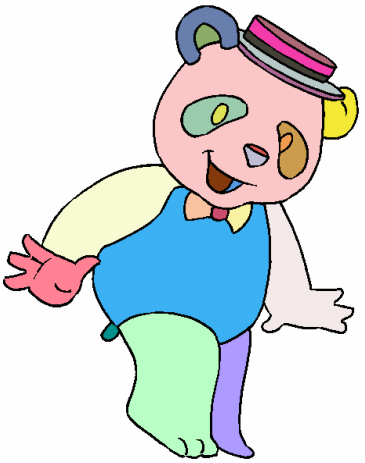

Figura 3.7: Rotulamento de uma imagem através de algoritmos de preenchimento.

os *pixels* contidos na região interior a essa curva. Executando-se esse algoritmo a partir de todos os *pixels* ainda não identificados de uma imagem I, uma coleção contendo todas as regiões de I será produzida. Seja N o número total de regiões e i o rótulo de cada região, ao final desse processo, a imagem I será idenficada por:

$$I = \bigcup_{1 \leq i \leq N} R_i \tag{3-1}$$

Esta abordagem possui um custo computacional eficiente $O(n)$, onde n é o número de *pixels* na imagem. Se a cada execução o algoritmo colorir os *pixels* identificados com uma cor diferente, um efeito semelhante ao da Figura 3.7 será encontrado.

Identificando o contorno de uma região

Uma região é uma união dos *pixels* pertencentes à borda e à sua área interna. Para cada região identificada na imagem, é possível encontrar o seu contorno calculando a curva que a envolve. Ao analisar todos os *pixels* que compõem uma região à procura de *pixels* de cor preta numa vizinhança 4-conectada, bordas internas e externa à região podem ser identificadas.

Através da identificação de cada região da imagem, é possível construir uma máscara para indicar sobre qual região a interpolação deve ser aplicada como mostra a Figura 3.8.

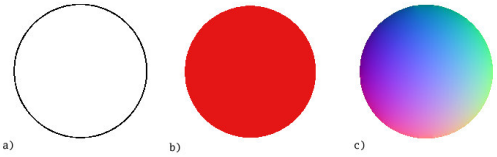

Figura 3.8: Máscara (em vermelho) indicando a região da imagem a ser interpolada.

3.8 Interpolação

Uma vez que os vetores foram estimados onde possível, um estágio de interpolação aplica interpolação esparsa para aproximá-los sobre a imagem restante. Esta etapa é baseada no trabalho de S. F. Johnston [25] para aproximação de vetores normais em imagens de múltiplos canais.

O sucesso deste trabalho depende de uma interpolação precisa dos valores através de um campo quando apenas pouco é conhecido em localidades esparsas. Ao invés de solucionar o problema através de complexos problemas de interpolação de normais como vetores unitários multidimensionais, as componentes normais N_x e N_y são interpoladas independentemente, e N_z é re-calculada para manter o vetor unitário. Assim, pixels com valores já conhecidos não são modificados; pixels com valores desconhecidos são relaxados através do campo usando o valor dos pixels vizinhos. Dado um campo de valores P a serem interpolados, e um campo de velocidade V inicializado com zero, cada iteração é calculada como sendo:

$$V'_{i,j} = d.V_{i,j} + K.(P_{i-1,j} + P_{i+1,j} + P_{i,j-1} + P_{i,j+1} - 4P)$$
$$P'_{i,j} = P_{i,j} + V'_{i,j}$$

Johnston [25] demonstra que valores ótimos para d e k variam. $d = 0.97$ e $k = 0.4375$ minimizam o tempo de convergência para um cálculo. Iterações são executadas até que a média-quadrada por pixel da velocidade alcance um erro aceitável de tolerância.

A Figura 3.9 (esquerda) mostra um quadro de uma animação. O mesmo quadro após o processo de interpolação é mostrado à direita onde as normais estão representadas no espaço RGB.

Escalamento em Z

Para ajustar a curvatura da região, Johnston [25] propõe 2 métodos. O primeiro troca cada vetor normal por um outro de uma esfera escalada

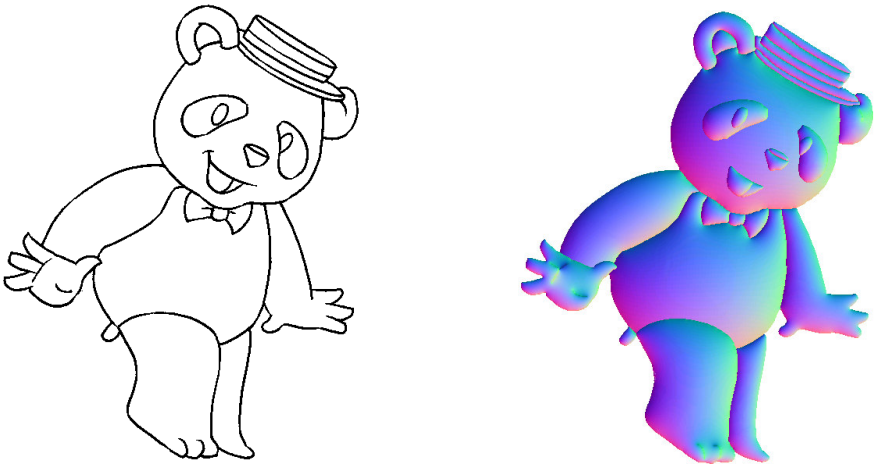

Figura 3.9: Interpolação. À esquerda, imagem original. À direita, normais no espaço RGB.

em z; o segundo método troca cada vetor normal por outro de uma porção de uma esfera escalada uniformemente.

Para mapear um campo de normais em uma esfera escalada em z por S, cada *pixel* deve ser calculado trocando-se (N_x, N_y, N_z) por:

$$\frac{(S \cdot N_x, S \cdot N_y, N_z)}{\|(S \cdot N_x, S \cdot N_y, N_z)\|}$$

Quando o fator de escala é menor do que 1, a região torna-se achatada, e quando é maior do que 1, a região torna-se pontuda como mostra a Figura 3.10.

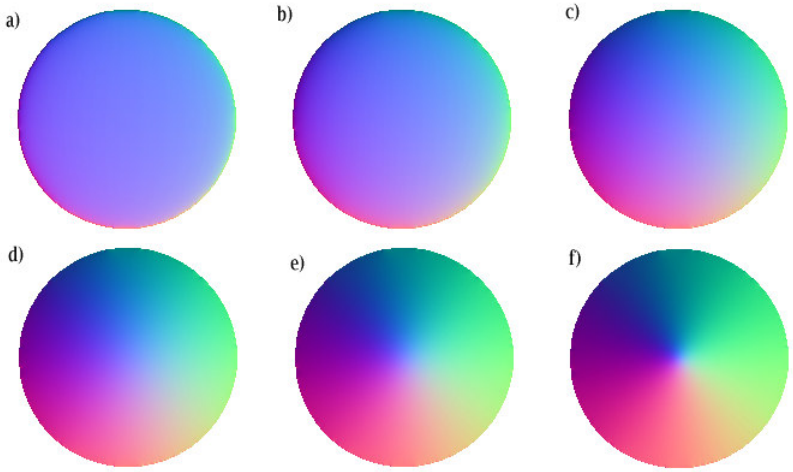

Figura 3.10: Normais mapeadas numa esfera escalada em z. a)$S = 0.25$ b)$S = 0.5$ c)$S = 1$ d)$S = 2$ e)$S = 4$ f)$S = 10$.

Em regiões onde houver um decaimento brusco, é preferível tornar as normais das arestas mais súbitas, em vez de manter $N_z = 0$ ao longo das arestas. O objetos visualizados em perspectiva apresentam ligeiramente, em

suas silhuetas, componentes z não-nulas. Por exemplo, a porção visível de uma esfera é uma parte da frente da esfera. Para remapear normais de um hemisfério em normais de um pedaço visível da esfera unitária com espessura S, N_x e N_y devem ser escalados por $\sqrt{S * (2 - S)}$ e N_z deve ser calculado de modo a manter o comprimento unitário. A Figura 3.11 mostra o resultado deste remapeamento para alguns valores de S.

Figura 3.11: Normais mapeadas numa esfera uniforme. $S = (0.125, 0.25, 1)$.

Os vetores normais mostrados na Figura 3.9 aproximam a geometria do urso para uma forma esférica dando-lhe uma aparência tridimensinal. Embora esta seja uma boa aproximação para a geometria do urso, algumas restrições devem ser observadas. Pode ocorrer o caso de um objeto estar ocludindo parte de um outro objeto; neste caso, as arestas do objeto ocludente influenciam o formato do objeto parcialmente ocludido, causando uma deformação neste último objeto. Isto pode ser observado, por exemplo, no caso da gravata sobre a barriga do urso. Uma possível solução para esse problema é o usuário escolher ignorar algumas arestas pertencentes à região, como por exemplo, ignorar os olhos quando calculando as normais da cabeça do urso. Uma outra possibilidade seria ainda o acréscimo de novas arestas, no entanto, tal abordagem não é prática.

Usando algoritmos baseados na imagem é possível aproximar superfícies normais diretamente evitando a transformação da cena para uma geometria 3D. As técnicas apresentadas neste capítulo são ideais para prototipagem de sombreamentos baseados em imagem. Para um maior controle e qualidade, renderizadores 3D devem ser utilizados.