Geração de ilustração a partir de modelo 3-D texturizado com imagem vetorial

Haroldo W. T. da Silva, Rafael B. Gomes e Luiz M. G. Gonçalves
Universidade Federal do Rio Grande do Norte
Avenida Senador Salgado Filho, 3000, CEP 59.078-970, Natal, RN, Brasil

***Abstract*—Even though there are many algorithms to generate illustrations from 3D models, few work with textured meshes, and even less output closed polygons. It is proposed, in this work, a technique to output non-photorealistic vector images that correspond to the vector texture mapping over screen projected 3D models. Basically, the Sutherland-Hodgman algorithm is adapted to cut geometric features from the texture and project them on the screen. This technique substantially increases the artist's control over the final picture appearance, when compared to methods that do not use textures, without loosing the resolution-independent characteristic.**

***Resumo*—Apesar de existirem vários algoritmos para gerar ilustrações a partir de modelos 3D, poucos lidam com modelos texturizados e menos ainda formam polígonos fechados. É proposta, neste trabalho, uma técnica para gerar imagens vetoriais não-fotorrealistas que correspondem ao mapeamento de texturas vetoriais sobre modelos 3D projetados na tela. Basicamente, o algoritmo de Sutherland-Hodgman é adaptado para recortar os objetos geométricos da textura e projetá-los na tela. Esta técnica aumenta substancialmente o controle do artista sobre a aparência da imagem final, quando comparada a métodos que não usam texturas, sem perder a característica de ser invariante à resolução.**

***Keywords*-textura vetorial; SVG; arte vetorial;**

I. Introdução

Muito embora uma grande parcela dos estudos em computação gráfica se dedique à imitação e reprodução do mundo real, imagens de natureza não-realista encontram seu lugar tanto em aplicações práticas quanto em estudos científicos. A necessidade de produzir ilustrações de natureza puramente artística recebe força, na era da informação, de programas que, baseando-se em padrões matemáticos, auxiliam ou completam o talento do artista, poupando tempo e garantindo resultados interessantes. Por outro lado, ainda que se deixe de lado o desejo de impressionar, as imagens vetoriais continuam sendo úteis, devido à sua natureza descritiva e flexibilidade. O ato de gerar uma ilustração vetorial a partir de informação objetiva, no entanto, não é uma tarefa trivial, pois requer um forte embasamento teórico e dedicação aos temas específicos.

O padrão SVG fornece uma solução para divulgação e visualização de arte vetorial na internet, sendo uma alternativa a ser explorada para a geração dinâmica de conteúdo. É preciso, no entanto, combinar a seriedade fria da geração procedural com o talento artístico humano, para maximizar a qualidade dos resultados.

Um modelo 3-D, gerado pelo homem, e uma imagem vetorial, também gerada pelo homem, podem ser combinados, computacionalmente, para gerar uma terceira imagem, também vetorial, que seria o correspondente vetorial da renderização desse modelo 3-D. O objetivo deste trabalho é justamente demonstrar como executar esse procedimento em SVG, usando uma adaptação do algoritmo de Sutherland-Hodgman. A principal inovação proposta é a geração de uma ilustração vetorial, invariante à resolução, que representa a visualização de uma malha de triângulos 3-D com mapeamento de uma textura, também vetorial. No artigo, é descrito como utilizar uma imagem vetorial diretamente como textura para uma ilustração que representa o objeto 3-D, sem rasterizar a textura, sem utilizar pipeline gráfico e tendo como resultado um gráfico vetorial, não uma matriz de pixels. Como citado, foi desenvolvido um método para utilizar imagens SVG como textura, recortando os caminhos através de uma versão modificada do algoritmo de Sutherland-Hodgman e mapeando-os na tela.

II. Trabalhos anteriores

Na literatura, é possível encontrar uma série de técnicas para gerar ilustrações a partir de objetos tridimensionais [1], [2], [3]. Rusinkiewicz et al. [4] faz uma compilação comparativa de diversas dessas técnicas, dando pleno foco ao desenho de linhas e estilização, mas não à geração de polígonos fechados. Karsch et al. [5] usam *snaxels* para definir polígonos fechados na imagem de saída, e aplica estilização tanto nos preechimentos quanto nos contornos. Além disso, o trabalho gera resultados no formato SVG.

Estudos envolvendo mapeamento de texturas vetoriais estão em menor número. A solução de Qin et al. [6] preocupa-se em gerar uma imagem raster a partir de texturas vetoriais, não tendo, assim, um resultado invariante à resolução. Trabalhos que gerem ilustrações invariantes à resolução a partir de texturas vetoriais são bem mais escassos. O estudo de Winkenbach e Salesin [7] demonstra como gerar ilustrações com estilo de traçado artístico a partir de texturas do mesmo tipo. Sua técnica, no entanto, também não envolve polígonos fechados.

III. Conceitos gerais

Basicamente, a técnica proposta no presente trabalho consiste em recortar os caminhos na imagem da textura, usando uma versão modificada do algoritmo de Sutherland-Hodgman [8], e transformar os caminhos resultantes para o espaço da tela.

Textura vetorial

O formato de imagem vetorial pode ser entendido como um conjunto de elementos formalmente definidos para descrever a forma e aparência de objetos geométricos planos. Uma imagem vetorial é um conjunto de instâncias desses elementos, usados para representar uma figura específica. Essa definição contrasta com a de uma imagem *raster*, que essencialmente é uma matriz de pixels, cada um indicando um valor de cor e eventualmente de transparência.

A visualização de uma imagem *raster* revela *aliasing* quando a imagem é ampliada, e distorções quando ela é rotacionada, fruto de sua essência discreta. Uma imagem vetorial, por outro lado, pode ser ampliada e rotacionada sem perda de sua qualidade visual ou de informação. Além disso, a natureza descritiva das imagens vetoriais permite que elas armazenem informação em um alto nível de abstração, relacionando-a com os aspectos visuais específicos de seus elementos, o que resulta em uma riqueza de dados impossível quando se tem apenas valores de cores.

O Padrão SVG

O formato de imagem utilizado para os testes deste trabalho foi o SVG. O padrão *Scalar Vector Graphics*, ou SVG, foi criado e é mantido pelo W3C como um formato aberto de imagem vetorial, com suporte a animação e *scripts* do usuário [9]. Tendo sido construído sobre o formato XML, conta-se com uma ampla disposição de ferramentas para lê-lo e manipulá-lo. Em um documento SVG, a informação visual é dada por primitivas geométricas descritas por elementos XML. Essas primitivas são organizadas em uma estrutura em forma de árvore, como é típico de documentos XML, e suas características visuais são definidas tanto por seus atributos individuais como pelas características do seu ramo específico na árvore do documento.

Matriz de transformação atual: Sendo primitivas geométricas, os elementos visuais de SVG têm atributos para especificar detalhes de sua topologia em termos numéricos. Um desses atributos é o *transform*, que permite especificar uma matriz de transformação, a qual irá modificar os aspectos geométricos na visualização do elemento. Essa matriz é sempre no formato dado pela equação (1).

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

A especificação SVG descreve meios de determinar os valores de a a e, seja diretamente ou por descrição de alto nível do tipo de transformação. As transformações dos elementos são aplicadas também aos filhos, por multiplicação à direita.

A transformação final de um elemento, resultante da multiplicação de sua matriz individual pelas de seus ancestrais, é chamada de *current transformation matrix*, ou CTM. Assim, um ponto X_e descrito no espaço de um elemento C, cujo ancestral direto é B, o qual tem apenas um ancestral, A, será mapeado para o ponto X_i da imagem como na equação (2):

$$X_i = M_A M_B M_C X_e \tag{2}$$

onde M_k é a matriz de transformação do elemento k. É importante enfatizar que *os pontos relacionados à geometria de um elemento são descritos no espaço do elemento e a posição desses pontos pode mudar na imagem.*

Caminho: O formato SVG disponibiliza um conjunto de tipos de elementos gráficos para compor as imagens. Alguns desses elementos são mais primitivos e outros são mais complexos, mas para os fins deste trabalho, nos concentraremos em uma primitiva chamada *caminho*. Um caminho SVG é uma sequência de comandos de traçado de desenho que orientam sobre a aparência de um certo elemento visual. Formalmente, um caminho C é uma sequência definida como $C = \langle f_0(t), f_1(t), \ldots, f_k(t) \rangle$, onde cada $f_k(t)$ é uma função paramétrica que mapeia t para um ponto na imagem. No desenho de um caminho, cada função é traçada em sequência, para $t = [0, 1]$. Se o caminho formar um contorno fechado, então ele poderá ter preenchimento. Um caminho pode ser um polígono fechado, mas também pode ser uma sequência de curvas bicúbicas de Bézier, formando uma figura fechada ou não. Este trabalho se limita a trabalhar com polígonos e sequências de segmentos retos, mas a técnica, em sua visão mais abstrata, pode ser aplicada a qualquer função paramétrica contínua no plano.

Modelo tridimensional

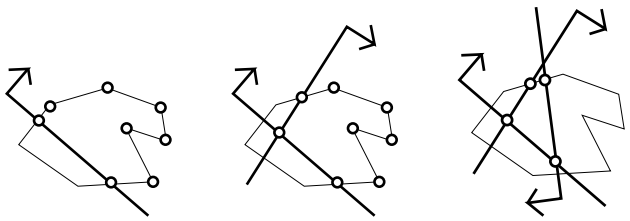
Para os fins deste trabalho, os termos malha de triângulos e modelo 3-D são considerados sinônimos. Cada triângulo no espaço tridimensional deve estar associado a um triângulo no espaço bidimensional, correspondente ao seu mapeamento de textura. Não há modificação na técnica se for utilizada mais de uma textura ao mesmo tempo, mas as descrições feitas aqui consideram que uma única textura vetorial está sendo mapeada sobre a superfície de um modelo 3-D. Espera-se que a malha seja contínua e fechada.

Visibilidade

A questão de quais triângulos da malha são ou não visíveis é importante, tanto do ponto de vista da corretude quanto da eficiência. Naturalmente, a aplicação da textura somente deve ser feita nos triângulos visíveis. Devido à natureza invariante à resolução da imagem de saída, algoritmos baseados em pixel, como Z-buffer e similares, não são adequados. Nenhuma estrutura de dados particular é proposta, mas é preciso eliminar os polígonos fora do volume de visualização e garantir que os triângulos mais próximos do observador sejam desenhados por último. A técnica foi desenvolvida tendo em mente uma malha contínua e fechada, e sem auto-interseções. Caso haja auto-interseções, é preciso recortar os polígonos da textura após a etapa final de projeção.

IV. Recorte de caminho

Nesta seção, é detalhado o algoritmo de recorte de polígonos proposto, que é uma adaptação do algoritmo de Sutherland e Hodgman [8]. Em sua versão original, o algoritmo é proposto para determinação de visibilidade, recortando os polígonos fora do volume visível no espaço 3D. No entanto, interessa



(a) Um plano de corte (b) Dois planos de corte (c) Três planos de corte

Figura 1. Algoritmo de Sutherland-Hodgman

aqui o recorte de polígonos bidimensionais, da textura vetorial, por um triângulo de corte também bidimensional, gerado a partir do mapeamento da textura do modelo 3-D usado.

O algoritmo de Sutherland-Hodgman foi escolhido porque não impõe restrições sobre a natureza dos polígonos a serem recortados, funcionando igualmente bem para polígonos côncavos ou convexos, sequências de vértices não-coplanares e até curvas paramétricas. A única restrição é que os polígonos de corte sejam convexos, mas isso não traz dificuldades, já que os polígonos de corte aqui utilizados são todos triângulos.

Cada polígono de corte é subdividido em planos de corte. Cada plano de corte será definido por seu vetor normal e um de seus pontos. Naturalmente, como o processo é feito no plano 2-D, poderia se esperar que os objetos de corte fossem retas, cada uma definida por dois pontos, e essa visão é reforçada pelo fato de que cada plano de corte é calculado a partir de uma aresta de um triângulo, mas essa representação não é eficaz, como veremos adiante.

Formalização

Um plano de corte P é formalmente representado por sua normal, $\mathbf{N}$ e por um de seus pontos, $\mathbf{R}$, ambos tratados como vetores bidimensionais. Cada plano de corte subdivide o espaço $\mathbb{R}^2$ em uma área chamada *visível* e outra, chamada *invisível*. Definimos que um ponto $\mathbf{Q}$ qualquer está na área visível se e somente se $\mathbf{N} \cdot \mathbf{Q} > 0$.

Se um ponto não está na área visível, então ele está sobre o plano ou na área não visível. Aqui nota-se a vantagem da representação escolhida para o objeto de corte, como um plano definido por uma normal e um ponto nele, em vez de uma reta definida por dois pontos. No segundo caso, seria preciso determinar valores adicionais para se determinar em qual área um ponto está.

A interseção das três áreas visíveis dos planos de corte corresponde ao interior do triângulo de corte. A idéia principal do algoritmo é que os segmentos do polígono a ser recortado são, em ordem, submetidos ao teste de intersecção com cada um dos planos de corte, sendo a saída de cada etapa repassada como entrada à etapa seguinte. Cada segmento k pode ser entendido como uma função paramétrica $f_k(t)$, com o formato da equação (3):

$$f_k(t) = (1-t)\mathbf{S_0} + t\mathbf{S_1} \tag{3}$$

Esse segmento traça uma linha reta entre um ponto de origem $\mathbf{S_0}$ e um ponto de destino, $\mathbf{S_1}$, para $0 < t \leq 1$. O ponto $\mathbf{X}$ em que o plano intersecta o segmento pode ser determinado pela equação (4).

$$\mathbf{N} \cdot (\mathbf{X} - \mathbf{R}) = 0 \tag{4}$$

Se há intersecção, então $\mathbf{X} = f_k(t)$ para um t específico, determinável pela equação (5).

$$t = \frac{\mathbf{N} \cdot \mathbf{R} - \mathbf{N} \cdot \mathbf{S_0}}{\mathbf{N} \cdot (\mathbf{S_1} - \mathbf{S_0})} \tag{5}$$

Há alguns detalhes importantes a serem ressaltados. Em primeiro lugar, a divisão não existe quando o segmento é ortogonal ao vetor normal do plano de corte, como esperado. Isso não significa que o segmento não pertença à área interna do triângulo de corte, ele apenas não intersecta esse plano em particular. Em segundo lugar, o valor de $\mathbf{N} \cdot \mathbf{S_0}$ é constante para cada plano de corte e pode ser registrado por performance. A saída do recorte, para cada segmento k, para cada plano de corte, deve conter os seguintes pontos, nesta ordem:

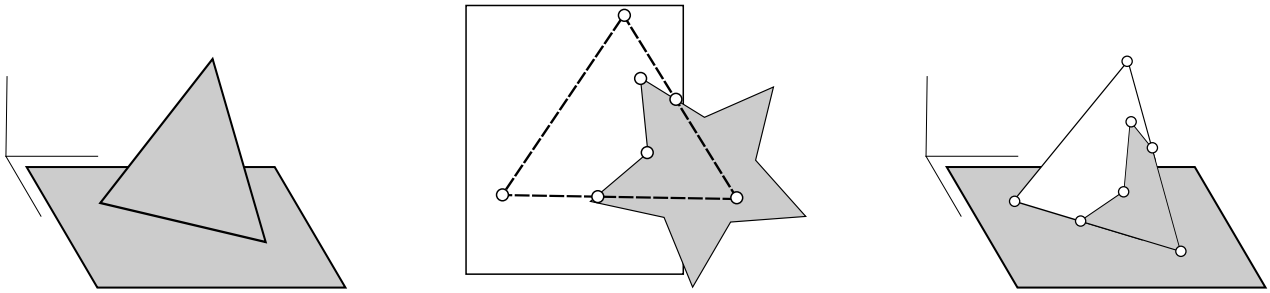
1) $\mathbf{S_0}$, se for o primeiro ponto do caminho atual e estiver na área visível;
2) $\mathbf{X}$, se há intersecção para $0 < t \leq 1$;
3) $\mathbf{S_1}$, se estiver na área visível;

A saída do algoritmo de corte, para um plano de corte e um caminho, ou é vazia, ou é outro caminho. Se a saída for vazia, então o caminho está completamente fora do triângulo de corte e o procedimento pode parar. Senão, o caminho-resultado deve ser submetido ao corte pelo plano seguinte, e depois pelo seguinte (Figura 1). O resultado final é um caminho que corresponde à intersecção entre o caminho original e o triângulo de corte.

Influência da CTM

Os triângulos de corte, e seus planos de corte, são descritos em coordenadas no espaço da imagem. Cada caminho, no entanto, está descrito por coordenadas no espaço do elemento, o qual está sujeito à sua matriz de transformação atual, a CTM. Em geral, tentar fazer o recorte sem levar em conta essa transformação irá produzir resultados incorretos. Uma abordagem que funcionaria, nesse caso, seria transformar todos os pontos de cada caminho antes do recorte. No entanto, uma idéia aindaa mais eficiente é transformar cada plano de corte com a inversa da CTM de cada elemento antes de iniciar o procedimento de recorte.

Assim, para recortar um elemento de CTM M_e, usando um plano de corte P descrito pela normal $\mathbf{N}$ e pelo ponto $\mathbf{R}$, é preciso gerar o novo plano de corte $\mathbf{P_t}$ com normal $\mathbf{N_t} = {M_e}^{-1}\mathbf{N}$ e ponto $\mathbf{R_t} = {M_e}^{-1}\mathbf{R}$. Para que sejam possíveis tais multiplicações, é preciso que os pontos e vetores sejam tratados com uma terceira coordenada. Essa terceira coordenada deve ser 1 nos pontos, para que sejam afetados pela translação, e 0 nos vetores, para que não o sejam. *Pode-se notar, no entanto, que a CTM de um elemento pode não ser invertível*. Nesse caso, resta apenas multiplicar cada ponto do caminho pela CTM antes do recorte.



(a) Triângulo no espaço 3D (b) Triângulo de corte sobre textura (c) Triângulo com textura vetorial

Figura 2. Projeção dos polígonos recortados

V. Projeção no espaço da tela

Uma vez recortados os polígonos, é preciso projetá-los no espaço da tela. Cada triângulo de corte corresponde a um triângulo da malha 3-D projetado na tela, então a questão se resume a encontrar a matriz P que transforma o triângulo de corte no triângulo 3-D projetado, e aplicar essa transformação nos polígonos recortados. Um triângulo T, entendido como um conjunto de três pontos em $\mathbb{R}^2$, pode ser convertido em uma matriz M_T como em (6).

$$M_T = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$${M_T}^{-1} = \frac{1}{x_1 y_2 - y_1 x_2} \begin{bmatrix} y_2 & x_2 & x_2 y_3 - y_2 x_3 \\ y_1 & x_1 & x_1 y_3 - y_1 x_3 \\ 0 & 0 & x_1 y_2 - y_1 x_2 \end{bmatrix} \tag{7}$$

Nesse formato, escolhido para manter coerência com as matrizes de transformação SVG, cada coluna recebe uma coordenada do triângulo. A inversa de uma matriz com esse formato, se existir, pode ser calculada como no trabalho de Sutherland (7).

Assim, sendo M_C a matriz correspondente ao triângulo de corte e M_S a matriz do triângulo projetado na tela, podemos encontrar $\mathbf{P} = M_S {M_C}^{-1}$. Uma vez determinada essa matriz, resta aplicá-la aos polígonos cortados, como demonstrado na Figura Figura 2.

VI. Conclusão

Uma grande vantagem das imagens vetoriais é o seu aspecto invariante à resolução. Isso significa que o mesmo arquivo, com o mesmo tamanho, pode ser renderizado em quaisquer níveis de detalhes sem perda de qualidade.

Este trabalho demonstra como aplicar mapeamento de textura vetorial a um modelo 3D sem rasterizar nem a textura e nem a imagem final. Esse procedimento, e o fato de que imagens vetoriais podem ser pequenas e ricas em detalhe ao mesmo tempo, abre portas para usos na web, por exemplo, na forma de renderização de cenas tridimensionais por demanda e visualização no navegador.

O formato SVG, no entanto, é muito flexível, e algumas nuances do seu uso como textura ainda não estão resolvidas. Deixamos para a continuação dos nossos trabalhos uma modificação do algoritmo de recorte para curvas Bézier quadráticas e cúbicas, bem como a otimização para animação.

Referências

[1] L. Markosian, M. A. Kowalski, D. Goldstein, S. J. Trychin, J. F. Hughes, and L. D. Bourdev, "Real-time nonphotorealistic rendering," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 415–420. [Online]. Available: http://dx.doi.org/10.1145/258734.258894

[2] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 848–855, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882354

[3] T. Judd, F. Durand, and E. Adelson, "Apparent ridges for line drawing," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276401

[4] S. Rusinkiewicz, F. Cole, D. DeCarlo, and A. Finkelstein, "Line drawings from 3d models," in *ACM SIGGRAPH 2008 Classes*, ser. SIGGRAPH '08. New York, NY, USA: ACM, 2008, pp. 39:1–39:356. [Online]. Available: http://doi.acm.org/10.1145/1401132.1401188

[5] K. Karsch and J. C. Hart, "Snaxels on a plane," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '11. New York, NY, USA: ACM, 2011, pp. 35–42. [Online]. Available: http://doi.acm.org/10.1145/2024676.2024683

[6] Z. Qin, M. D. McCool, and C. Kaplan, "Precise vector textures for real-time 3d rendering," in *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, ser. I3D '08. New York, NY, USA: ACM, 2008, pp. 199–206. [Online]. Available: http://doi.acm.org/10.1145/1342250.1342281

[7] G. Winkenbach and D. H. Salesin, "Computer-generated pen-and-ink illustration," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 91–100. [Online]. Available: http://doi.acm.org/10.1145/192161.192184

[8] I. E. Sutherland and G. W. Hodgman, "Reentrant polygon clipping," *Commun. ACM*, vol. 17, no. 1, pp. 32–42, Jan. 1974. [Online]. Available: http://doi.acm.org/10.1145/360767.360802

[9] D. Jackson and C. Northway, "Scalable vector graphics (svg) full 1.2 specification," World Wide Web Consortium, Working Draft WD-SVG12-20050413, April 2005.