Real-Time 3D Artistic Rendering System

Tong-Yee Lee, Shaur-Uei Yan, Yong-Nien Chen, and Ming-Te Chi

Department of Computer Science and Information Engineering,
National Cheng-Kung University, Tainan, Taiwan, Republic of China
tonylee@mail.ncku.edu.tw

Abstract. This paper presents an artistic rendering system for generating 3D images of Chinese paintings using graphics hardware. The user can adjust suitable parameters flexibly to generate different brush styles as his/her hobby, and see rendering results in real time. In this system, we propose a hardware-accelerated method to draw Chinese painting strokes efficiently along visible silhouettes. Three-dimensional texture and multi-texture from normal graphics hardware is used to speed up generating various brushes with Chinese painting stylized strokes. The features of the traditional Chinese painting such as ink diffusion and moisture effects are simulated. Several examples of aesthetically pleasing Chinese-paintings rendered from 3D models are demonstrated using the proposed method.

1 Introduction

In the past, most non-photo-realistic rendering (NPR) researches focus on the western painting styles such as pen-and-ink, watercolor, hatching and so on. However, few works in NPR are about Chinese paintings and most of them focus on simulating delicate effects of brush, black ink and papers. Furthermore, most of them are 2D Chinese drawing works and computationally expensive for real-time applications. These researches are interested in its simulated quality rather than in its processing time. However, when generating the scene of Chinese painting style in games or virtual environment, the real-time performance is required and we cannot use these previous works directly. In this paper, we present a real time NPR system for generating 3D Chinese paintings. The system pipeline consists of four stages and it is illustrated in Fig. 1.

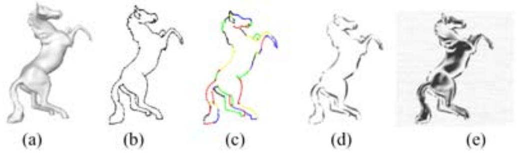

Fig. 1. System overview. (a) Input model, (b) visibility testing, (c) visible segment linking, (d) stroke placement and (e) interior shading. In (c), we color each linked segment

2 Related Work

Strassmann models hairy brushes in his 2D oriental black-ink painting system [1]. This work represents each stroke with a cubic spline and renders the stroke using polygons with texture. Lee [2] designs a 3D brush model with elastic bristles that respond elastically to the force exerted by an artist against the paper. To simulate realistic diffuse effects of blank-ink paintings, Guo et al. consider the sorbency of paper, the liquid density and flow [3]. Zhang et al. [4] propose to use cellular automation-based simulation of ink behavior to render 3D trees in sumie style. Way et al [5] propose a method of synthesizing rock texture in Chinese landscape painting. Later, they further develop methods to simulate 3D Chinese painting trees using silhouettes with texture strokes [6]. Chan et al. [7] exploit existing software packages such as Maya and RenderMan to create 3D Chinese painting animation. Chu et al. [8] develop a system utilizing Haptic input device to track 3D brush movement to give more accurate brush simulation. Yeh et al. [9] propose a set of algorithms to render 3D animal models in Chinese painting style. To shade the interiors of animals, several basic image processing techniques such as color quantization, ink diffusion and box filtering are used.

3 Stylizing Silhouettes with Brush Strokes

3.1 Stroke Paths and Widths Generation

The idea for drawing view-dependent silhouettes of 3D model with stylized strokes is popular in NPR. We adopt Isenberg et al.' approach [10] to find visible silhouettes and concatenate silhouette segments into long stroke paths. Before applying stylizations onto stroke paths, control points along paths need to be interpolated using cubic spline to smooth the curvature of stroke paths. To make a stylized stroke path, we need to grow various widths at control points. In traditional Chinese painting, the brush width starts with thin stroke and gradually grows to thick stroke, and turns back to thin stroke as the brush stroke progresses. Yeh et al. [9] assign stroke width based on the order of control points only and potentially generate less smooth transition between different brush stroke widths when a long brush path contains fewer control points. To solve this problem, we consider distance between control points as another parameter to control the width of brush stroke. We use Eq. (1) and (2) to compute the width. In Eq. (1), ncp represents the total number of control points on a given stroke path. Eq. (2) represents the width of the stroke at a given control point i, which is 0 at both starting and ending points of the brush path. Eq. (1) returns the width to add or to subtract given the condition at the ith control point, where Vlength[i] is distance between the ith and the (i-1)th control points and width_step is a predetermined width step value. We demonstrate an example to compare [9] and our approach in Fig. 2. The proposed approach yields better visual stroke appearance than [9].

$$add[i]=\begin{cases} +VLength[i]*width\_step, & if\ \ i<\frac{1}{2}\times ncp\ \ and\ \ width[i]<MAX\_WIDTH \\ -VLength[i]*width\_step, & if\ \ i\geq\frac{1}{2}\times ncp\ \ and\ \ width[i]\geq 0 \\ 0, & else \end{cases} \tag{1}$$

$$width[i]=\begin{cases} 0 & ,if\ \ i=0\ \ or\ \ i=ncp \\ width[i-1]+add[i] & ,else \end{cases} \tag{2}$$

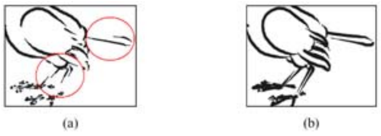

Fig. 2. Different stroke widths generated by different methods. Left: stroke generated by [9] Right: stroke generated by our system

3.2 Brush-Strokes in Chinese Painting Style

A brush consists of many bristles. In a microscopic view, when a single bristle draws on the paper, the effects it can produce different ink shades such are dark, light, dry and wet when ink diluted with water; with different pressure, direction in brush stroke, different ink tones can produce millions of variations of touches on the paper. Various brush-strokes are usually used to represent different texture of the subjects in Chinese painting. To simulate different shades of ink, we define the term ”pattern” to refer to the ink traces on the paper left by a bristle. The pattern setup is illustrated in Fig. 3 by combining an intensity map and an opacity (i.e., alpha) map. We can use 2D texture to store each pattern. When painting an absorbent paper, an artist can control the water content in the brush to make ink look sear, soggy or wet. The opacity map is used to control water content and therefore it is called a moisture map, too. The intensity map is used to control ink shades such as dark and light. With different combinations of bristle patterns, different style of brush strokes can be produced; Fig. 4 shows example of our simulated brush strokes in the style of flying-white (fei-bei) technique and slanted brush technique in Chinese painting. To efficiently generate the brush stroke patterns in real time, the hardware-accelerated 3D volume texture and multi-texture techniques are used in our system. Intensity changes are mapped to a 3D texture level to encode intensity change into the

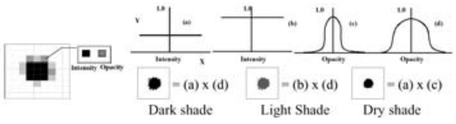

Fig. 3. Brush stroke setup on paper

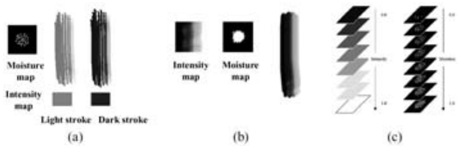

Fig. 4. (a) flying-white (fei-bei) brush path (Left). (b) slanted brush stroke path. (c) 3D texture representing intensity and moisture maps for brush strokes

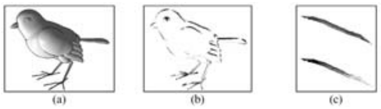

Fig. 5. By placing light, stroke brightness distribution can be controlled(Middle). Brush moisture effect(Left)

third dimension of 3D texture. Moisture changes are also mapped to another 3D texture level to account the moisture change in the third dimension as well. See figure 4(c) for a visual representation of this idea. The user just needs to prepare two sets of predetermined intensity and moisture maps. Then, we load these two sets into 3D volume texture. At running time, when an arbitrary intensity value is specified, corresponding 2D texture pattern can be interpolated efficiently from two neighboring intensity value automatically by the graphics hardware, thus a intensity map of the brush at the given intensity value is obtained. Similarly, the moisture map of a given moisture value can be computed in this hardware accelerated manner. With OpenGL extension [11], it allows us to enable multi-texture technique to combine the two maps into a new brush pattern. In this way, the system can deliver brush pattern with desired intensity and moisture value very fast using hardware accelerated 3D volume texture and multi-texture techniques. Next, we will give details about how to provide both intensity and moisture value at running time.

In Section 3.1, control points along every stroke path contain its original 3D coordinates and normal vector. Given a light source, we can compute a light vector from a control point to a light source. The intensity value of each control point is computed by the dot product of these two vectors and this dot product is normalized to the range of (0,1), so that all paths have different intensity values and are influenced by the lighting condition; distribution of bright and dark stroke can be gathered by giving different configuration to lighting as illustrated in Fig. 5. When an artist actually draws on paper, the moisture of the brush changes from wet to dry from the beginning of a stroke to the end of it. In order to simulate this effect, the control points in the beginning of each path are assigned a predetermined amount of moisture, and the moisture level in consequent control points drops as the distance from the first control point increases. See Fig. 5(c) for an example of the moisture effect. After the intensity and moisture value are found at each point along a brush path, we can use them as the third dimension of 3D volume texture to fast compute corresponding intensity and moisture stroke "pattern".

Table 1. Performance breakdown for Fig. 6

Models	Teapot	Sparrow	Horse
Num of vertices	530	4502	6918
Num of faces	992	9184	13832
Visibility test	71.8 fps	54.9 fps	48.6 fps
Path linking	70.5 fps	51.4 fps	43.9 fps
Stroke placement	58.1 fps	20.9 fps	20.4 fps
Interior shading	54.0 fps	20.2 fps	20.2 fps

4 Interior Shading

In this section, we present a method to draw colors in the interior area of models. The goal of this method is to fast simulate ink color change from dark to light like ink diffusion in Chinese painting. We use Eq. (3) to simulate this change in the interior of models.

$$Opacity = A * \cos^{n}(\frac{\theta}{W}) \tag{3}$$

Where θ is the angle between a vertex normal and a light vector from a vertex to light, A, n and W are constants to control shape of this function. To implement this simulation, the fragment shader provided by Nvidia's CG language [12] is used to do per pixel opacity value calculation using Eq. (3). By using the calculated opacity values and user-defined ink intensity for shading the object, the change of brightness can be seen after we enable blending. The brightness change can be seen as ink diffusion in Chinese ink painting. To avoid the dull appearance of uniform color distribution, noise functions such as Perlin noise can be used to add some randomness. In the next Section, several interesting results will be demonstrated to verify the proposed method for interior shading. In contrast to other ink diffusion approach [3, 4], the proposed method computes very fast but it yields not bad results.

5 Experimental Results

The experimental setup in this paper is a program written in C++ and OpenGL using Microsoft Visual C++ 6.0 compiler running on an Intel Pentium 4 R 2.2Ghz machine with Microsoft Windows 2000. Graphics card is Nvidia GeForce FX5900 with 256MB frame buffer. In Table 1, three models are used to test rendering speed measured in frame per second (fps) at 800x 600 screen resolutions. Because our system is implemented in four different stages, we list the frame rate (i.e., fps) at different stages. We can see the current performance bottleneck is limited by the stroke placement stage. The rendering performance we achieve is fast enough for user interaction in real time. Fig. 6 shows rendered results for Table 1. More results are demonstrated in Fig. 7.

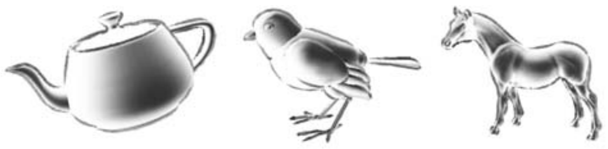

Fig. 6. We show three rendered results used in Table 1

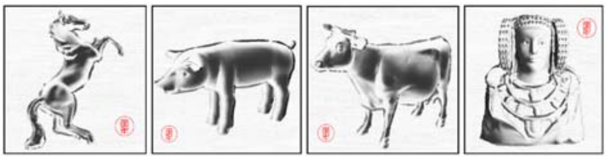

Fig. 7. More rendered results

6 Conclusion and Future Work

This paper presents a real-time NPR rendering system to generate traditional Chinese ink paintings for 3D models. The proposed method is accelerated by the normal graphics hardware. This method consists of four main steps: 1) visibility testing, 2) path linking, 3) stroke placement and 4) interior shading. For the stroke placement, the 3D volume texture and multi-texture techniques are used to fast compute ink and moisture information. We also attempt to simulate ink diffusion effect to paint the interiors of the models. As a result, many aesthetically pleasing Chinese-paintings rendered in real-time from 3D models are demonstrated using the proposed method. There are many possible future work can be further explored based on our current work. For example, we plan to consider motion issue in NPR such as sparrow jumping or waving its wings created by using bone and skin deformation techniques. In this situation, we need to consider the stroke coherence problem. Without treating well this issue, it is very easy to create popping effect during animation or deformation. Another our research direction is to how to express model metamorphosis [13, 14] or human face [15] in Chinese painting style or in western painting style [16].

Acknowledgement

This paper is supported by the National Science Council, Taiwan, Republic of China, under contract No. NSC-93-2213-E-006-026 and NSC-93-2213-E-006-060.

References

1. Strassmann, S., "Hairy brushes," Proc. SIGGRAPH 86, 20(4): 225-232, August 1986.
2. J. Lee, "Simulating oriental black-ink painting," Computer Graphics and Applications, IEEE, vol. 19(3), pp. 74-81, May-June 1999.

3. Q. Guo and T. Kunii, "Modeling the Diffuse Painting of Sumie'," Modeling in Computer Graphics (Proc. IFIP WG5.10), T. Kunii, ed., Springer-Verlag, Tokyo, 1991,pp. 329-338.
4. Q. Zhang, Y. Sato, J. Takahashi, K. Muraoka and N. Chiba, "Simple cellular automation-based simulation of ink behavior and its application to Suibokuga-like 3D rendering of trees," Journal of Visualization and Computer Animation, 1999.
5. Way, D. L., and Shih, Z. C. "The Synthesis of Rock Texture in Chinese Landscape Painting," Computer Graphics Forum, Vol. 20, No. 3, pp. C123-C131, 2001.
6. Way, D. L., Lin, Y. R. and Shih, Z. C. "The Synthesis of Trees Chinese Landscape Painting Using Silhouette and Texture Strokes." Journal of WSCG, Vol. 10, No. 2, pp.499-506, 2002.
7. C. Chan, E. Akleman, and J. Chen, "Two methods for creating Chinese painting," Proceedings.10th Pacific Conference on, October 2002, pp. 403 - 412.
8. N. S.-H. Chu and C.-L. Tai, "An efficient brush model for physically based 3d painting," in Proceedings of 10th Pacific Conference on Computer Graphics and Applications, 2002, 2002, pp. 413-421.
9. Jun-Wei Yeh, Ming Ouhyoung, "Non-Photorealistic Rendering in Chinese Painting of Animals," Journal of System Simulation, Vol. 14, No. 6, 2002, pp. 1220-1224.
10. T. Isenberg, N. Halper, and T. Strothotte, "Stylizing silhouettes at interactive rates: From silhouette edges to silhouette strokes," in Computer Graphics Forum, Proceedings of Eurographics 2002, vol. 21, September 2002, pp. 249-258.
11. http://www.opengl.org
12. http://developer.nvidia.com/page/cg_main.html
13. Tong-Yee Lee, P.H Huang, "Fast and Institutive Polyhedra Morphing Using SMCC Mesh Merging Scheme," IEEE Transactions on Visualization and Computer Graphics, Vol. 9, No. 1, pp. 85-98, 2003.
14. Chao-Hung Lin, Tong-Yee Lee, "Metamorphosis of 3D Polyhedral Models Using Progressive Connectivity Transformations," IEEE Transactions on Visualization and Computer Graphics, Jan./Feb. Issue, Vol. 11, No.1, pp. 2-12, 2005
15. Tong-Yee Lee, Ping-Hsien Lin, Tz-Hsien Yang, "Photo-realistic 3D Head Modeling Using Multi-view Images," in Lecture Notes on Computer Science (LNCS) 3044, Springer-Verlag, pp. 713-720, May 2004.
16. Ming-Te Chi, Tong-Yee Lee, "Stylized and Abstract Painterly Rendering System Using a Multi-Scale Segmented Sphere Hierarchy," to appear in IEEE Transactions on Visualization and Computer Graphics.