Stroke-based Real-time Rendering of Ink Wash Style for Geometric Models

Tian-Chen Xu*[1] Li-Jie Yang†[1] En-Hua Wu‡[1, 2]
[1]Faculty of Science and Technology, University of Macau, Macao, China
[2]State Key Lab of CS, Institute of Software, Chinese Academy of Sciences, Beijing, China

Figure 1: *Rendering results on a character in complex topology with different predefined rendering parameters: texturing in thick ink (left most); texturing in light ink (the 2nd on the left); texturing with coloring (2 on the right)*

Abstract

This paper presents a novel approach for real-time rendering of ink wash style on geometric models. We first create an ink footprint model in particle form, then extract classified strokes from ordinary 3D models by utilizing the existing geometry information, and finally combine the strokes and the ink to generate an immersive painted image. By taking advantage of parallel processing on GPU, consequently, the rendering of geometric models with realistic features of Chinese ink painting can be performed on a consumer-level PC in real-time.

CR Categories: I.3.3 [Computer Graphics]: Picture / Image Generation—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture ;

Keywords: Non-photorealistic Rendering, Ink Wash Painting, Real-time Rendering

1 Introduction

Ink wash painting was originated in China, whose goal is to present the soul of the subject instead of simply describing its appearance and shape. Artists usually express the diverse properties of the subject with hair brush only through various compounds of water and ink. This particular painting style possesses irreplaceable position and immeasurable value in the art world. The art had been also introduced to Korea and Japan, and developed a few new styles (Sumi-e etc.) by integrating their own culture and innovation, thus becoming a very popular and typical class of art in eastern painting.

Although it is impossible for using computer simulated image to replace the position of real artwork for artistic appreciation, since ink wash painting has quite high visual value, it is still attractive to integrate this style into computer applications. In order to generate ink wash painting animations for video games and films with traditional procedure, artists have to paint every image for each frame, thus tremendous time and energy are required. For instance, a masterpiece of Chinese ink painting animation "The Cowboy's Flute" (1963) lasts only 21 minutes, but it took 2 years to complete the work. At present, this work has been reproduced by rendering on 3D models for only one month. In this regard, it is demanding to introduce computer rendering techniques in order to improve the efficiency for producing large number of ink wash paintings in animations.

In current applications, it is still difficult to accurately convey the features of strokes when rendering on the 3D objects. For example, in the game "Okami" by Capcom Co., Ltd, a very successful attempt to integrate ink painting elements into video game, the Japanese painting style is highlighted in interactive system. However, many 3D models in the game are still rendered as "3D models", which does not seem to be painted by strokes on 2D paper. Actually, an ideal ink painting style contains the perception of hierarchy but without full stereoscopic. This paper is an attempt of step forward to achieve this goal.

Ink wash painting has various painting techniques of drawing. Apart from others, four primary techniques are well known and well concluded: contouring, texturing (cun), coloring, and dotting. Contouring is to sketch the silhouette of objects with thick ink, and texturing, called "cun" in Chinese, is to present the grains and cracks on object surfaces with dry brush, which express the perceptions of stiffness and softness. Coloring is to shade the object to reflect the illumination with light ink, and dotting is to paint the mosses and trees on the rock or mountain abstractly to make the work full of vitality. In this paper, we will concentrate on these techniques to generate immersive strokes in virtue of our algorithms and the power of modern graphics pipeline.

*e-mail: TianchenX@siggraph.org
†e-mail: YangLijie81@gmail.com
‡e-mail: EHWu@umac.mo

ACM Reference Format Tian-Chen Xu, Li-Jie Yang, and En-Hua Wu. 2012. Stroke-based real-time ink wash painting style rendering for geometric models. In *SIGGRAPH Asia 2012 Technical Briefs* (SA '12). ACM, New York, NY, USA, Article 19, 4 pages. DOI=10.1145/2407746.2407765 http://doi.acm.org/10.1145/2407746.2407765

SIGGRAPH Asia 2012, Singapore, November 28 – December 1, 2012.

2 Related Work

For ink simulation and rendering, Xu et al. [2004], Chu and Tai [2004] proposed virtual brush models, and simulated ink-paper interaction to produce brush footprint for ink rendering. Later, Chu and Tai [2005] presented an approach of real-time ink dispersion in absorbent paper based on fluid dynamics for their brush model. Although this method can generate realistic ink by physical simulation and works wonderfully for the painting utility, it is not suitable for real-time 3D model rendering, since we expect to simulate the final ink effect rather than the painting procedure. In iR2s: Interactive Real photo to Sumi-e [Xie et al. 2010], ink rendering along the strokes is also mentioned for transforming photos into ink paintings with six prepared footprint textures which takes up much texture buffers when it is used in real-time rendering engine.

With the development of graphics pipeline, real-time NPR is not only limited to the style conversion based on image processing. At present, the generation of line drawings is one of the main areas of real-time NPR. DeCarlo et al. [2003] proposed suggestive contours for conveying shape, which covers the basic silhouette extraction with more extended geometrical details in good view dependency. On the other hand, as GPU technology evolves, NPR research takes advantages of GPU parallel computation to accelerate rendering algorithms. For example, Chen et al. [2005] developed a hardware accelerated Chinese landscape painting rendering system.

3 Ink Simulation

3.1 Ink structure & I/O format definition

In our method, the ink along a stroke consists of particles, and each particle has its geometric properties and color information. Naturally, the final data output from graphics pipeline is the color values each pixel, thus we define the output format as the $RGBA$ values for a certain fragment of the particle, among which we output the intensity value into the Alpha channel. The intensity value is influenced by the intrinsic ink density, brush footprint, ink dryness, and paper property:

$$\alpha = (kF + k_{diff}F_{diff})I - k_{dry}\xi D \quad (1)$$

where I, F's, D denote the intrinsic ink density, footprint value, and dryness respectively; k's are the coefficients of blending weight; the quantities with subscript $diff$ represent the corresponding values after diffusing.

In calligraphy and ink painting works, there is a particular ink effect called "fly-white". When a dry brush swept on the paper rapidly, some fragmentary areas are not colored with ink. In order to simulate this effect, as is shown in Eqn. (1), the last term is for fly-white control, and ξ is a random number to adjust the threshold.

Besides, for the geometry properties, we define two values: footprint size and particle rotation, which are determined by brush pressure and stroke orientation. For their detailed usage, we will explain in the next subsection. Therefore, after compositing all factors discussed above, we define the input format for particles as (P, A, D, I) where the four components denote the input brush pressure, orientation, dryness, and the intrinsic density respectively.

3.2 Footprint generation

The brush footprint is constructed on each particle, including footprint texture and mask. As is shown in Figure 2, the mask is generated procedurally: we first model the whole brush footprint mask, and then cut out part of the mask based on the pressure value

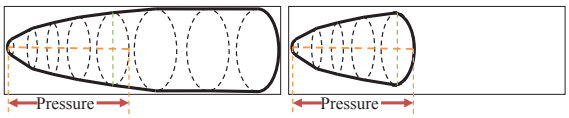

Figure 2: *The whole brush footprint mask (left); The brush footprint mask (right)*

$P \in [0, 1]$ from (P, A, D, I). After the projective construction of brush texture and mask, the particle is rotated around the footprint tip according to the stroke orientation A denoted in radian angle. Thus, the complete procedure of footprint generation from the 4D data structure $PADI$ is sketched as below. For the diffusing operation, since the ink diffusing is influenced by the paper blocking factor, which is an intrinsic property of paper, here we use noised sampling to simulate it.

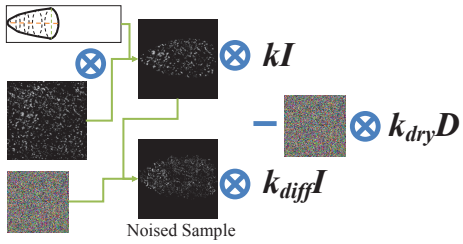

Figure 3: *The procedure of footprint generation based on Eqn. (1) in the last section*

4 Stroke Extraction

In this section, we focus on the four main technical steps introduced before to extract strokes from ordinary 3D model with vertex position, normal, tangent and texture coordinates information. All the steps output curves with stroke data in the same $PADI$ structure as we defined in the previous section.

4.1 Contouring

Contouring is to stroke the silhouette of the object, thus we first have to find the silhouette. There are two types of silhouette: the intrinsic edges of the object, and the outlines with viewpoint dependency. For the former type, we traverse the index list of vertices during preprocessing, and mark the indices whose corresponding edge is not shared more than one triangle. For the latter type, we must mark them out during the real-time rendering. For each triangle, we first calculate $\vec{V} \cdot \vec{N}$ for each vertex, where $\vec{N}$ and $\vec{V}$ denote the normal vector and represents the direction vector from the current vertex to the camera position in world space respectively. Then, as is shown in Figure 4, referring to the silhouette detection technique by visibility [Hertzmann and Zorin 2000], if the $\vec{V} \cdot \vec{N}$ values on the edge are in contrary sign, we solve the equation $[(1-\lambda)\vec{N}_1 + \lambda\vec{N}_2] \cdot [(1-\lambda)\vec{V}_1 + \lambda\vec{V}_2] = 0$ to compute the interpolation parameter λ, and than calculate the positions in view-projection space, texture coordinates, and other data by linear interpolation.

Now, we expect to determine the brush pressure and the orientation of the silhouette segment. Here, we refer to environmental mapping. First, we map the object into a sphere in the modeling space, and then project into viewing space without translation. Thus, we have defined a circle. After computing the angle as the pressure parameter from the circle parametric equation clockwise, we output the final pressure value P by trigonometric function. Simultaneously, now we can calculate the orientation A denoted in radian

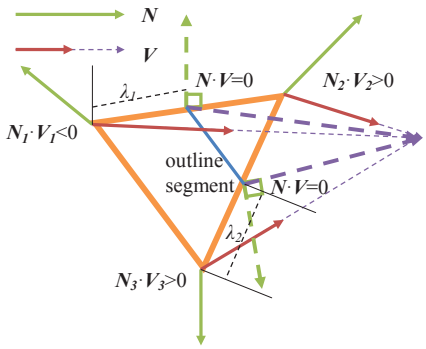

Figure 4: *Silhouette seeking for each triangle*

angle from the vertex positions of the corresponding silhouette segment, and the direction is determined by the pressure parameter which suggests the direction along the stroke.

In addition, the dryness and ink density are predefined constants in shader. According to the basic knowledge of ink wash painting, the ink density of contouring tends to be a heavy value, and the dryness is assigned with a medium value.

4.2 Texturing

Texturing reflects the material properties, thus we could utilize the texture mapping as usually done in ordinary photorealistic rendering. After sampling from diffuse texture, we convert it into grey value G, then we compute the pressure, ink density and dryness using following equations respectively:

$$P = (k_{Pscale}\rho f(\vec{V}\cdot\vec{N}))^{k_{Pexp}} \quad (2)$$

$$I = (k_{Iscale}\rho)^{k_{Iexp}} s \quad (3)$$

$$D = (k_{Dscale}Gf(\vec{V}\cdot\vec{N}))^{k_{Dexp}} \quad (4)$$

where $\rho = 1-G$, k's are adjusting parameters, and $f(x)$ is the pressure function determined by edge closing term $1-\vec{V}\cdot\vec{N}$, since in texturing technique of ink wash painting, the ink is stroked and decays along the direction away from the silhouette. Here, s is a mask value of stripes, sampled using intrinsic texture coordinate with repeat whose texture is shown on the right.

Figure 5: *Stroke mask*

Furthermore, as is mentioned before, the orientation of the stroke faces to the direction away from the silhouette.

4.3 Coloring

Since coloring is related to the illumination, here we take advantage of Phong Illumination model [1975]. After calculating the illumination as is in ordinary photorealistic rendering using Phong model, we represent it into grey value G. Then, we compute the pressure and ink density similarly to texturing, but without the stripe mask. Due to the fact that the texturing intensity are much stronger than the coloring strokes, and coloring often attaches to the areas of texturing in many ink wash paintings, we only fill the area that has not been textured, or the density of texturing is zero.

Besides, the orientation of stroke is to point along the gradient of texture coordinates, and the dryness is predefined as well, which tends to be a low value according to the painting knowledge.

4.4 Dotting

Dotting is usually to represent entire objects like trees and mosses which are much smaller than the attached objects like rocks and mountains. Thus, we just need to simply generate one particle with thick ink density and low dryness, if the size of the entire object in view-projection space is less than the threshold. Besides, the pressure is defined by the object size, and the orientation is set to zero.

5 Implementation & Results

Our rendering system is implemented in DirectX 11, and our testing machine is equipped with Graphic Card Nvidia® Geforce GT240M and Intel ® Core™2 Duo CPU P7450.

5.1 Rendering pipeline

We make a multi-pass rendering pipeline, but the total rendering has only two passes. In the first pass, we render the objects into an $RGBA$ color buffer for $PADI$ structure and a depth buffer for occlusion culling. Then, we extract the silhouette strokes. In the vertex shader, we simply pass the vertex data in view-project space. The silhouette strokes are constructed using the method mentioned in the previous section in the geometry shader. $PADI$ values from the prior stage are output in the pixel (fragment) shader. Then, we render the object again for the texture and coloring strokes: in the same pixel shader, we sample the texture for texturing and compute the Phong Illumination for coloring, and then calculate the $PADI$ values using our method. Moreover, for dot strokes, the tiny objects are replaced by one pixel individually.

In the second pass, we send vertices into the input assembler to fill the frame-buffer, whose number equals the number of output pixels under current resolution. In higher resolution requirements for anti-aliasing, we distribute multiple number of vertices as supersampling. Then, we read the buffer in the vertex shader using the Vertex-Texture-Fetch (VTF) technique. In the geometry shader, we generate particles from points, and those vertices with zero-D (ink density) values are discarded by geometry shader. In the pixel shader, we calculate the final output color by the computation shown in Figure 3.

The complete pipeline is visualized as shown in Figure 6 below.

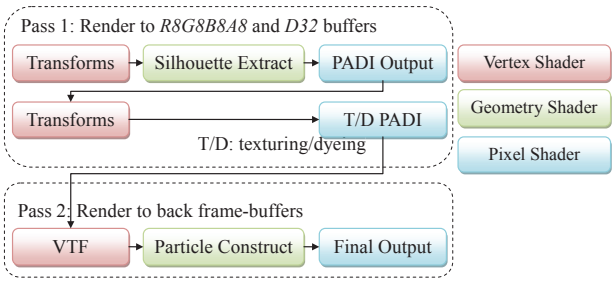

Figure 6: *Rendering pipeline*

5.2 Results

In this section, we first give the result images rendered by our method, as shown in Figure 1 and the figures below.

As mentioned in the previous subsection, we output curves in the first pass, and then expand them into particles. Thus, each pixel output from the first pass is sent as a vertex in the second pass.

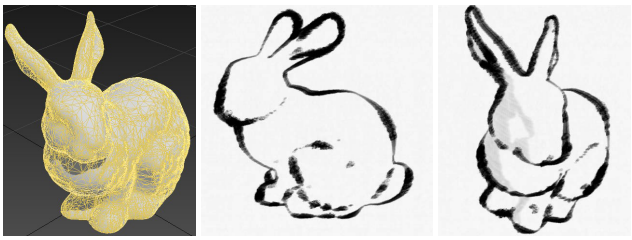

Figure 7: *Experiment on results on the bunny: input 3D mesh shown with wire-frame (left); our rendering results (middle & right)*

Figure 8: *Experiment on a deformable character in complex topology: input 3D mesh shown with wire-frame (left); input 3D mesh shown in photorealistic rendering (middle); our rendering result (right)*

Therefore, the efficiency of the second rendering pass is tightly related to the required output resolution rather than the complexity of the scene. Here, we particularly count the rendering speed for various resolution, as shown in the table below. Care must be taken for that, sometimes, we need more vertices for anti-aliasing by super-sampling, which is not aimed at wiping out staircase-shaped edges during rasterization as if it were in the photorealistic rendering, but it is for the sake of keeping the curve continuity when sampling the stroke curve texture during VTF.

Resolution	Mesh vertices	Super sampling	FPS
640 × 480	34191	1x	230 - 250
640 × 480	34191	2x	130 - 150
800 × 600	34191	1x	170 - 190
800 × 600	34191	2x	75 - 90
1024 × 768	34191	1x	90 - 110
1024 × 768	34191	2x	35 - 50

Table 1: *Rendering speed statistics for various resolutions*

6 Conclusion

In this paper, we present an approach to generate images of ink wash style for 3D models. In order to generate the flexible and immersive ink effect, we defined a 4D data structure $PADI$ to represent the abstracted ink data, and explored the methods for constructing curve-form strokes around the primary ink wash techniques (contouring, texturing, coloring, and dotting) by focusing on their painting features. Furthermore, we devised a simple rendering pipeline to implement our method in an optimized way, and generated some results with rich perception of ink wash painting.

Our future work mainly focuses on the ink generation and improves the method of brush pressure definition by fast tracing the stroke path. For ink effect, we expect to find a physically based approach efficiently for real-time 3D model rendering.

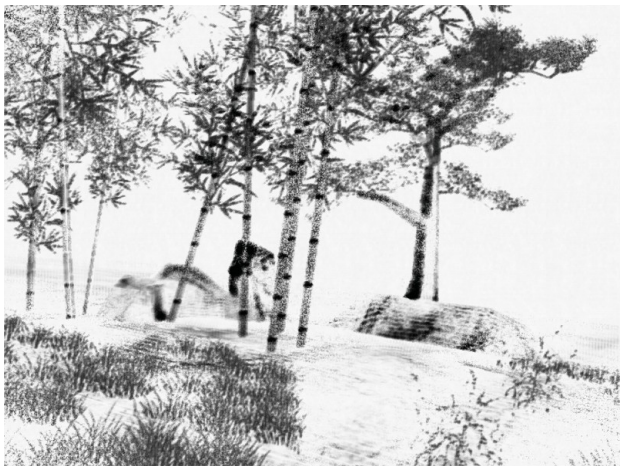

Figure 9: *Scene rendering of ink wash style*

Acknowledgements

The authors would like to thank the associate editor and all the referees for their constructive comments to improve the manuscript. This research has been supported by National Basic Research 973 program of China (2009CB320802, 2011CB302801), National Natural Science Foundation of China (60833007) and the research grant of University of Macau.

References

CHEN, W., ZHANG, H., AND YU, J. 2005. Hardware accelerated chinese landscape painting rendering. *J. of Computer-Aided Design & Computer Graphics 17*, 2427–2432.

CHU, N. S. H., AND TAI, C.-L. 2004. Real-time painting with an expressive virtual chinese brush. *IEEE Comput. Graph. Appl. 24*, 5 (Sept.), 76–85.

CHU, N. S.-H., AND TAI, C.-L. 2005. Moxi: real-time ink dispersion in absorbent paper. *ACM Trans. Graph. 24*, 3 (July), 504–511.

DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3 (July), 848–855.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 517–526.

PHONG, B. T. 1975. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June), 311–317.

XIE, N., LAGA, H., SAITO, S., AND NAKAJIMA, M. 2010. Ir2s: interactive real photo to sumi-e. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ACM, New York, NY, USA, NPAR '10, 63–71.

XU, S., TANG, M., LAU, F. C. M., AND PAN, Y. 2004. Virtual hairy brush for painterly rendering. *Graph. Models 66*, 5 (Sept.), 263–302.