# Hardware-Accelerated Sumi-e Painting for 3D Objects

Joo-Hyun Park, Sun-Jeong Kim, Chang-Geun Song[1], and Shin-Jin Kang[2]

[1] Department of Computer Engineering, Hallym University, South Korea
[2] Department of Games, Hongik University, South Korea

**Abstract.** Brushwork and ink dispersion make it difficult to render 3D common objects in the style of sumi-e painting. We use sphere mapping with brush texture and an image processing techniques to simulate brushstrokes and ink dispersion. The whole process is implemented in shaders running on Graphics Process Unit (GPU) that allows fast and high-quality rendering 3D polygonal models in the style of sumi-e painting. We show several results which demonstrate the practicality and benefits of our system.

## 1 Introduction

The sumi-e is an East Asian type of brush painting also known as ink and wash painting. Only black ink – the same as used in East Asian calligraphy – is used, in various concentrations. It is non-photorealistic rendering (NPR) which stands in contrast with to conventional graphics rendering methods of photo-realistic. The recent tendency of NPR system is simulating painting style and natural media, e.g. pen and ink, watercolor, charcoal, pastel, hatching, etc. About sumi-e paintings in NPR, many researches of 2D drawing systems have been shown. In these areas, the delicate simulations of brush, black ink and paper are presented, and a 2D image of sumi-e painting is generated accepting the hand drawing of the users.

There have been a number of systems for sumi-e painting brushwork and real-time NPR rendering. Early efforts in sumi-e painting focused on a brushwork simulation. Strassmann[11] swept a one dimension texture to show shading tone. Pham[9] modeled brushstrokes based on variable offset approximation of uniform cubic B-splines. Using the theory of elasticity, Lee[7] modeled a brush as a collection of rods with homogenous elasticity along the entire brush. Way[12] presented a method of synthesizing rock texture in Chinese landscape painting.

With development of GPU technologies, hardware-accelerated rendering skills began to be adapted to NPR system. Kang[4,5] modeled hardware-accelerated rendering algorithm for generating sumi-e painting in real-time from 3D meshes. Chu[2] worked on simulating real-time ink dispersion in absorbent paper using a fluid flow. Yuan[14] developed a GPU-based rendering and animation system for automatically generating Chinese painting cartoon from a set of mesh models.

G. Allen et al. (Eds.): ICCS 2009, Part II, LNCS 5545, pp. 780–789, 2009.

In NPR, many systems have addressed real-time NPR rendering. Majumder[8] implemented real-time charcoal rendering applied with contrast enhanced operators by using hardware-accelerated bump mapping and Phong shading. Lake[6] presented a method for cartoon rendering suits for programmable pipeline. Praun[10] introduced tonal art map and showed that it permitted real-time rendering of stroke-based texture for hatching rendering. And he also suggested hardware hatching system with Webb[13] using volume rendering and pixel shading. Kalnins[3] described a way to interactively render stylized silhouettes of animated 3D models with robust frame-to-frame coherence. Chi[1] developed a system for 3D NPR stylized and abstract painterly rendering using a multi-scale segmented sphere hierarchy.

In this paper, we present an interactive system to render 3D objects in the style of sumi-e painting. For burshstrokes we use sphere mapping with brush texture which represents brush style and can be changed for drawing a silhouette in various brush styles. For the interior of 3D models, we shade it using tone texture which can be modified to widen or narrow blank spaces in the painting. To simulate ink dispersion, we use an image processing technique which computes the weighted average of $k$-texel-away neighbors and then raises to the $\alpha$-th power. Two parameters $k$ and $\alpha$ are used to control the range and density of spreading. Finally we mix Xuan paper image with the result above to enhance the aesthetic sense. Our system enables users to interactively control all steps by changing brush texture, the parameter of tone texture, the value of spreading range and density, filtering techniques, and Xuan paper image. The whole rendering process is implemented in shaders running on GPU that allows fast and high-quality rendering. Our contributions are the follows:

- We propose two methods to simulate brushstorkes and ink dispersion. One is sphere mapping with brush texture so that brush pattern is mapped on a silhouette. The other is an image processing technique of the filtering controlled by the parameters of spreading range and density.
- We build an interactive rendering system by providing users with as many parameters as possible. As a result, user can draw sumi-e painting as their favors by determining the values of intuitive parameters.

## 2 Sumi-e Painting Algorithm

Our real-time system has three steps to render 3D models in the style of sumi-e painting: *silhouette outlining*, *interior shading*, and *paper effect*. (Fig. 1) In the first step, for simulating brushstrokes a brush pattern is mapped onto a silhouette of an object using sphere mapping. In the next step, the interior area of the object is shaded based on diffuse reflection and color obtained from object texture. The color of a pixel is determined using the tone texture for sumi-e painting. In the last step, image processing techniques enable to produce paper effect like ink dispersion and mixing background pattern.

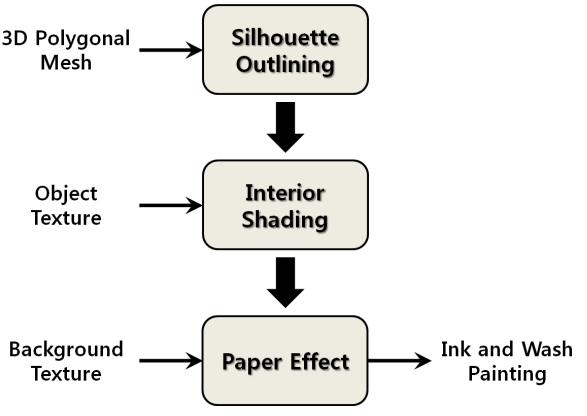



**Fig. 1.** System overview

### 2.1 Silhouette Outlining

A silhouette is a view of an object or scene of the outline and a featureless interior. A silhouette edge on a 3D object projected onto a 2D plane (display plane) is the set of points whose outwards surface normal is perpendicular to the view vector. A vertex $v$ is named a *silhouette vertex* if its normal vector $\mathbf{N}_v$ is almost perpendicular to the view vector $\mathbf{V}$.

$$0 \leq \mathbf{N}_v \cdot \mathbf{V} \leq \epsilon$$

where $\epsilon$ is the threshold of silhouette extraction and represents the width of silhouette. The value of $\epsilon$ can be interactively assigned by users in our system. The larger the number of $\epsilon$ is, the thicker the width of silhouette is. If the vertex $v$ is a silhouette vertex then its color is black. Otherwise its color is white.

One of difficulties in rendering 3D objects in the style of sumi-e painting is to render stylized silhouettes with brushstrokes. Unfortunately the scheme above cannot simulate brushstrokes. To achieve brush styles, we use sphere mapping with the brush texture following Kang's approach [4,5]. Sphere mapping can be accelerated by current hardware and has an advantage of not requiring addition calculation for silhouette detection. Also it can show various silhouette drawing effects easily by changing brush texture image.

In view coordinate system, we denote the vector from the vertex to the camera as $\mathbf{v}$, normalized to $\hat{\mathbf{v}}$. Since the computation is performed in view space, the camera is located at the origin and $\mathbf{v}$ is equal to $-\mathbf{p}$, where $\mathbf{p}$ is the position of the vertex in view space. The vertex normal $\mathbf{n}$ is transformed to view coordinates, becoming $\hat{\mathbf{n}}$. The reflected vector $\mathbf{r}(r_x, r_y, r_z)$ can be computed as:

$$\mathbf{r} = 2(\hat{\mathbf{n}} \cdot \hat{\mathbf{v}})\hat{\mathbf{n}} - \hat{\mathbf{v}}$$

We define:

$$m = 2\sqrt{r_x^2 + r_y^2 + (r_z + 1)^2}$$

(a) extracted silhouette (b) brushstrokes (c) brush texture

**Fig. 2.** A silhouette (a) is changed into a silhouette (b) using sphere mapping with the brush texture (c)

Then the texture coordinates $(t_u, t_v)$ are calculated as:

$$t_u = \frac{r_x}{m} + \frac{1}{2}, \quad t_v = \frac{r_y}{m} + \frac{1}{2} \tag{1}$$

Fig. 2(b) shows the silhouette produced by sphere mapping with the brush texture Fig. 2(c) . As we mentioned before, we can draw silhouettes with various brushstrokes by changing brush texture.

### 2.2 Interior Shading

Our approach to interior shading is similar to cartoon shading done on the GPU. First, we create a grayscale tone texture that contains the different shade intensities we desire. Fig. 3 shows the tone texture that we use in the rendering system. The tone texture intensity must increase from left to right. To smooth the abrupt transitions between shades, we blur the tone texture using Gaussian function.

Then at each pixel, we perform the standard diffuse calculation dot product to determine the cosine of the angle between the normal vector $\hat{\mathbf{N}}$ and the light vector $\hat{\mathbf{L}}$, which is used to determine how much light the pixel receives:

$$s = \hat{\mathbf{N}} \cdot \hat{\mathbf{L}}$$

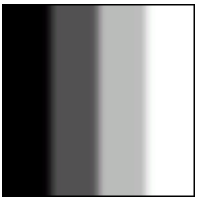


**Fig. 3.** Tone texture

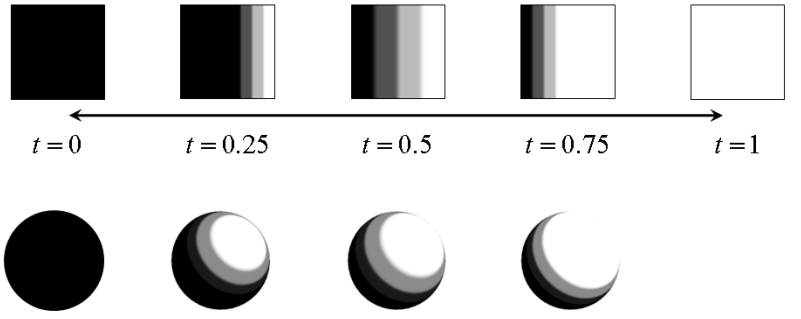



**Fig. 4.** The width of white shade in the tone texture is determined by the parameter $t$ and simultaneously the region of blank spaces in the painting is determined

If $s < 0$, that implies the surface receives no light. Therefore if $s < 0$, we let $s = 0$. So $s \in [0, 1]$.

Now, we use $s$ to scale our color vector $(r, g, b, a)$ got from object texture so that pixel colors are darken based on the amount of light that they receive:

$$diffuseColor = s(r, g, b, a)$$

Instead of using only $s$ as the $u$ texture coordinate for the shade texture, we compute the luminance of darken pixel color and use it as the $u$ texture coordinate for the tone texture. Therefore the texture coordinates $(u, v)$ are calculated as:

$$u = \min(s(0.3r + 0.59g + 0.11b), 1), \quad v = 0.5 \tag{2}$$

In the sumi-e painting, the concepts of implication and simplicity result in remaining a lot of blank spaces. In other words, many parts of the inside of an object are usually omitted. Our interactive rendering system enables to widen or narrow blank spaces by changing continuously the ratio of white shade to total tone texture. Fig. 4 shows that the parameter $t$ determines the width of white shade in the tone texture. If $t > 0.5$, then the area of white shade is relatively larger than that of black shade. It makes wider blank spaces in the interior of an object than those at $t = 0.5$. Otherwise, the area of white shade is relatively smaller than that of black shade. It also makes narrower blank spaces in the interior of an object than those at $t = 0.5$. If $t = 1$, then all the interior of an object is painted with the white color. In Fig. 4 the bottom row shows the examples of interior shading without the silhouette for a sphere model using the corresponding tone texture.

### 2.3 Paper Effect

After processes of silhouette outlining and interior shading, the color of a pixel is computed by the multiplication of silhouette and interior colors. We render

not to the screen, but to a texture $T$ which is prepared for image processing and whose resolution is same as that of the rendering window.

$$T(i,j) = \mathit{Pixel}(i,j) = \mathit{silhouetteColor}(i,j) \otimes \mathit{interiorColor}(i,j)$$

where the $\otimes$ symbol denotes component-wise multiplication.

To simulate ink dispersion, we use an image processing technique to compute the weighted average of $k$-texel-away neighbors from $T(i,j)$ in texture space and then draw it.

$$e(i,j) = \sum_{x=-1}^{1} \sum_{y=-1}^{1} T(i+kx, j+ky) G[x+1][y+1] \tag{3}$$

where $k$ is the range of spreading effect and $G[x][y]$ denotes a $3 \times 3$ Gaussian filter. If $k = 1$, then the spreading effect is same as the result image after a $3 \times 3$ Gaussian filtering. The larger the value of $k$ becomes, the further the ink is dispersed. In our rendering system, the value of $\alpha$ as well as $k$ can be interactively assigned. The input parameter $\alpha$ plays a role of the density of the spreading effect:

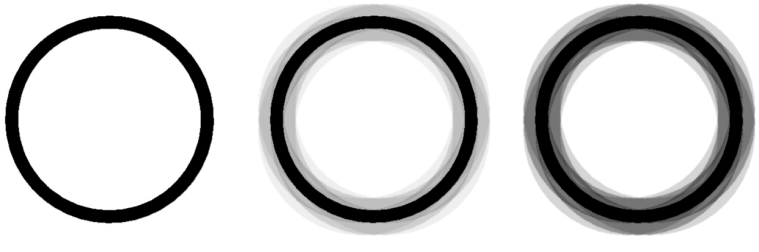
$$\mathit{filteredColor}(i,j) = \mathrm{pow}(e(i,j), \alpha) \tag{4}$$

where the function $\mathrm{pow}(b,n)$ returns $b^n$. Because $e(i,j) \in [0,1]$, the larger the number of $\alpha$ is, the darker the shade of the spreading effect is. Fig. 5(a) is a simple silhouette of a sphere model without interior shading. Using Equation (3), the silhouette color is spreading $k$ texels away (Fig. 5(b)). If the value of $\alpha$ becomes greater in Equation (4), the spreading effect becomes stronger (Fig. 5(c)).

Finally Xuan paper image is transformed into background texture and mixed with the result above by multiplication.

$$\mathit{Pixel}_{\mathit{final}}(i,j) = T(i,j) \otimes \mathit{filteredColor}(i,j) \otimes \mathit{backgroundColor}(i,j)$$

Because the end result of our rendering system is stored in the texture, we render it onto the screen-aligned billboard whose size is same as that of the rendering window.



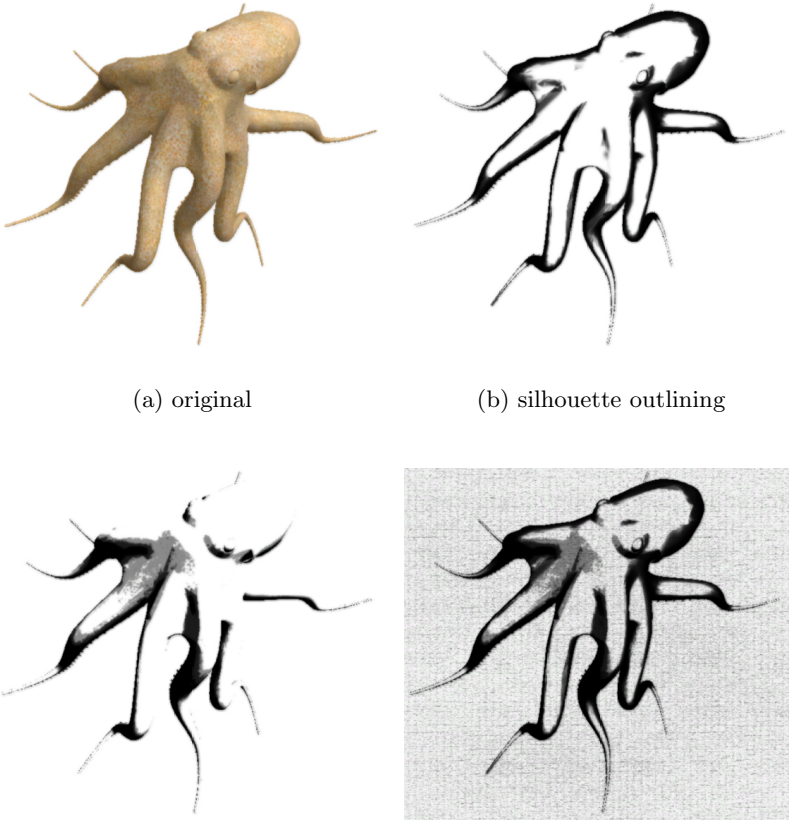
(a) silhouette (b) ink dispersion (c) high density

**Fig. 5.** Ink dispersion (b) of a silhouette (a) is simply simulated and darkened because of high density (c)

## 3 Implementation and Results

We have implemented an interactive rendering system using DirectX 9 and HLSL (High-Level Shading Language). The whole process is developed in shaders running on GPU.

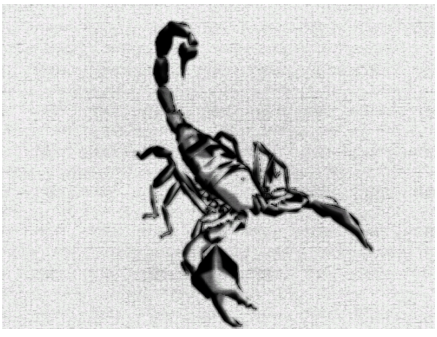
*GPU Processing:* Our algorithm entirely utilizes programmable GPU vertex and pixel shaders. Because all steps use texture mapping (brush texture, tone texture, and background texture), most operations are implemented in pixel shader. In vertex shader, the position and normal vector of a vertex in world space are transformed into the view coordinates, and texture coordinates pass through it. In pixel shader, silhouette extraction, diffuse reflection, brush texture mapping, interior shading, Gaussian filtering, pulp and spreading effects, and mixing background texture are worked on.



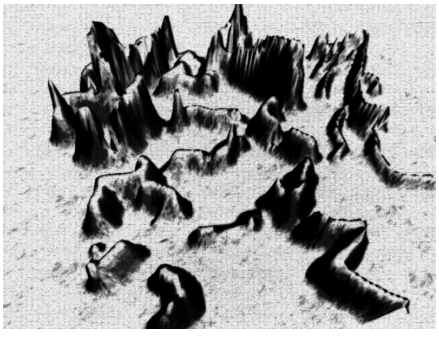
(a) original (b) silhouette outlining

(c) interior shading (d) paper effect + (b) + (c)

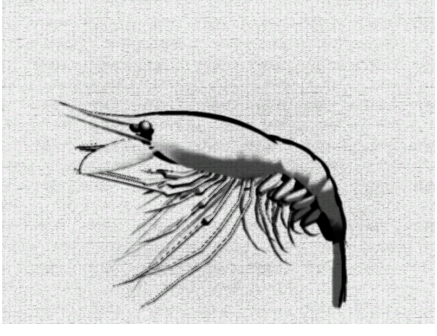
**Fig. 6.** Sumi-e painting of a octopus model



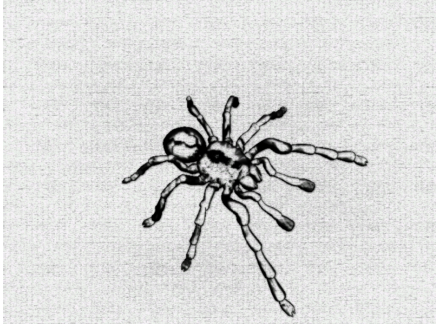
(a) scorpion



(b) terrain



(c) prawn



(d) spider

**Fig. 7.** Sumi-e paintings generated by our system

*Performance:* We conducted performance test on a machine with AMD Athlon 64 Dual Core Processor 6000+ CPU and a Geforce 8600GTS GPU with 512MB video memory. The performance data are shown in Table 1. Since half the processing time is consumed in the filtering step, the performance is independent of the number of vertices or triangles. The size of rendering window is the most important factor in the performance. The resolution of rendering window in Table 1 is $800 \times 600$. When the resolution is $400 \times 300$, the average of rendering performance is $800 \sim 850$ fps. When the resolution is $1920 \times 1200$, the average of rendering performance is $100 \sim 150$ fps.

*Results:* Fig. 2.3 shows an original model and results of each step. Fig. 7 shows sumi-e paintings of various 3D objects. Our input is X mesh files and dds image files for texture. Also we use 3ds MAX as the modeling tool and export models to X files. Users interactively control the brushstroke, shade of interior, spreading effect by filtering technique, and background paper in order to draw sumi-e paintings as their favors.

**Table 1.** Performance of our system

| Object | Vertices | Triangles | Frame/sec. |
|---|---|---|---|
| Octopus | 64,014 | 60,892 | 452 |
| Scorpion | 7,188 | 10,000 | 498 |
| Terrain | 16,641 | 32,768 | 363 |
| Prawn | 6,316 | 7,292 | 485 |
| Spider | 31,467 | 62,301 | 454 |

## 4 Conclusion

This paper presented an interactive system for rendering 3D objects in the style of sumi-e painting. For burshstrokes, we used sphere mapping with brush texture. We could draw a silhouette in various brush styles by changing this brush texture. For the interior of 3D models, we shaded it using tone texture which was modified to widen or narrow blank spaces in the painting. To simulate ink dispersion, we used an image processing technique which computes the weighted average of $k$-texel-away neighbors and then raises to the $\alpha$-th power. Two parameters $k$ and $\alpha$ were used to control the range and density of spreading. Finally we mixed Xuan paper image with the result above to enhance the aesthetic sense. Our system enables users to interactively control all steps by changing brush texture, the parameter of tone texture, the value of spreading range and exponent, filtering techniques, and Xuan paper image. The whole rendering process is implemented in shaders running on GPU.

In future work, we want to apply our rendering system to real-time game like cartoon rendering is already used in games. To do this we try to keep the coherence of silhouette for animating characters and devise rendering method for nature phenomena such as water and smoke in the style of sumi-e painting.

## References

1. Chi, M., Lee, T.: Stylized and abstract painterly rendering system using a multiscale segmented sphere hierarchy. IEEE Transactions on Visualization and Computer Graphics 12(1), 61–72 (2006)
2. Chu, N., Tai, C.: MoXi: real-time ink dispersion in absorbent paper. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005) 24(3), 504–511 (2005)
3. Kalnins, R., Davidson, P., Markosian, L., Finkelstein, A.: Coherent stylized silhouettes. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003) 22(3), 856–861 (2003)
4. Kang, S.-J., Kim, S.-J., Kim, C.-H.: Hardware-accelerated real-time rendering for 3D sumi-e painting. In: Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P. (eds.) ICCSA 2003. LNCS, vol. 2667, pp. 599–608. Springer, Heidelberg (2003)
5. Kang, S.-J., Kim, C.-H.: Real-time 3D sumi-e painting. In: ACM SIGGRAPH 2003 Conference Abstracts and Applications (Technical Sketch) (July 2003)

6. Lake, A., Marshall, C., Harris, M., Blackstein, M.: Stylized rendering techniques for scalable real-time 3D animation. In: Proceedings of NPAR 2000: The 1st International Symposium on Non-Photorealistic Animation and Rendering, June 2000, pp. 13–20 (2000)
7. Lee, J.: Diffusion rendering of black ink paintings using new paper and ink models. Computers & Graphics 25(2), 295–308 (2001)
8. Majumder, A., Gopi, M.: Hardware accelerated real time charcoal rendering. In: Proceedings of NPAR 2002: The 2nd International Symposium on Non-Photorealistic Animation and Rendering, June 2002, pp. 59–66 (2002)
9. Pham, B.: Expressive brush strokes. CVGIP: Graphical Models and Image Processing 53(1), 1–6 (1991)
10. Praun, E., Hoppe, H., Webb, M., Finkelstein, A.: Real-time hatching. In: Proceedings of SIGGRAPH 2001: The 28th Annual Conference on Computer Graphics and Interactive Techniques, August 2001, pp. 581–586 (2001)
11. Strassmann, S.: Hairy brushes. Computer Graphics (Proceedings of SIGGRAPH 1987) 20(4), 225–232 (1986)
12. Way, D., Shih, Z.: The synthesis of rock textures in chinese landscape painting. Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001) 20(3), 123–131 (2001)
13. Webb, M., Praun, E., Finkelstein, A., Hoppe, H.: Fine tone control in hardware hatching. In: Proceedings of NPAR 2002: The 2nd International Symposium on Non-Photorealistic Animation and Rendering, June 2002, pp. 53–58 (2002)
14. Yuan, M., Yang, X., Xiao, S., Ren, Z.: GPU-based rendering and animation for Chinese painting cartoon. In: Proceedings of GI 2007: Graphics Interface, May 2007, pp. 57–61 (2007)