한국컴퓨터그래픽스학회 Vol. 15, No. 2, P. 19~34

수묵화 렌더링 시스템

정규만 이승용°

대구대학교 정보통신공학부, 포항공과대학교 컴퓨터공학과°

kyuman.jeong@gmail.com, leesy@postech.ac.kr°

Oriental Black Ink Rendering System

Kyuman Jeong Seungyong Lee°

Daegu University, POSTECH°

요 약

이 논문에서는 삼차원 메쉬를 이용한 자동화된 수묵화 렌더링 시스템을 소개한다. 먼저 수묵화의 특징을 분석하고 이를 바탕으로 수묵화 렌더링 시스템을 위한 요구사항을 제시한다. 또한 수묵화를 그릴 때 사용되는 중요한 단계들을 살펴본다. 이러한 분석에 기반하여 본 논문에서 제시하는 시스템은 다음과 같은 세 가지 레이어로 나누어진다: 중요선 레이어, 내부 칠하기 레이어, 미디어 레이어. 본 논문의 가장 큰 특징은 사실적인 이미지 생성과 실시간 렌더링이라는 두 가지 요구사항을 모두 만족시킬 수 있는 새로운 방법을 제시한다는 점이다. 이 시스템은 몰골법, 백묘법, 구륵법 등 수묵화의 대표적인 세 가지 스타일을 쉽게 생성할 수 있다. 본 논문의 연구 결과는 컴퓨터 게임과 가상 환경 등 실시간 특성이 중요한 요소가 되는 어플리케이션에 바로 적용될 수 있을 것으로 예상된다.

Abstract

This paper presents an automatic rendering system for 3D meshes which generates images in the style of oriental black ink painting. In this paper, we analyze the characteristics of black ink painting and present some requirements for our black ink rendering system. Then we survey the steps used by artists to create black ink paintings. Based on the analysis of drawing steps, we propose a black ink rendering system which reproduces the key features of black ink painting through three sub-systems: the feature line layer, the interior shading layer, and the media layer. The main contribution of this paper is to introduce new methods which generate realistic and natural results very fast while any simulation methods are not used to satisfy real-time performance. We describe our implementation of each of these, and demonstrate our results in the major three styles of black ink painting: the outline style, the spontaneous style, and the outline-spontaneous style. Our system renders 3D models of moderate complexity at interactive frame rates. As a result, we expect that our system can be directly applied to real-time applications such as computer games and virtual environments.

키워드: 비사실적 렌더링, 수묵화 렌더링

Keywords: Non-photorealistic rendering, Black ink rendering

1. Introduction

Recently non-photorealistic rendering (NPR) has attracted much attention in computer graphics society [1, 2]. While photorealistic rendering focuses on generating images similar to real photographs through physically-based simulation, NPR targets on creating expressive, painterly, and interpretative images of objects without attempting to mimic the reality. These days, NPR widens its application areas from art to entertainment such as computer games and virtual reality. Many NPR techniques have been developed to produce the aesthetic effects of various painting styles in object ren-

dering. However, these techniques mainly deal with the western painting and drawing styles, e.g., water color [3], pencil rendering [4], and hatching [5].

Oriental black ink painting is an interesting unique art form, which has a long history of more than 3,000 years [6]. It has prominent characteristics that differentiate it from other painting styles, such as delicate gray tones, impressive textures with varying tones, the simplicity of using only a few strokes, and the beauty of blank [1].

To support the production of images in the black ink painting style, drawing systems have been developed with simulation of the physical tools [7, 8, 9, 10]. Although the systems can generate impressive images, the quality of the result depends on the human expertise and the simulation process may take much computation. For automatic generation of images in the black ink painting style, rendering techniques for 3D objects have been developed, where no simulation is used to speed up the rendering [11, 12, 13]. However, in the techniques, the characteristics of black ink rendering are too much simplified and may not be properly reflected on the resulting images.

In this paper, we present an oriental black ink rendering system for 3D meshes, which we call *OBIRS*. The system generates images of input 3D meshes in the style of oriental black ink painting. According to our analysis of the drawing steps and the painting styles of black ink painting, we found that the structure of the system is similar to existing NPR systems. OBIRS consists of three layers: feature line, interior shading, and media layers. However, the characteristics of black ink painting make a difference. Therefore, we analyze the characteristics of black ink paining to deduce requirements which should be satisfied. To satisfy the requirements, we propose some new techniques. Since any time-consuming simulation methods are not used in the system, it can render an input model of moderate complexity in real-time. The system can also support rendering with different styles of black ink painting. The resulting images demonstrate that the system can generate images similar to real black ink paintings.

Because the proposed system runs in real-time, it can be applicable to real-time applications. Nowadays NPR techniques are widely used for virtual environments. Klein *et al.* [14] presented a real-time NPR system for rendering architectural interiors. In the computer game industry, cartoon style rendering is used for games, e.g., *Tales of Windyland* and *Need for Speed: Underground*, because it is familiar to users and can also provide higher frame rates. For virtual environments and computer games, a rendering system must run in real-time without user intervention. Hence, the rendering system presented in this paper can be a good option for such applications. Our system can immediately replace the current renderer in computer games or virtual environment applications. In addition, OBIRS can be used to generate high quality animation of input 3D meshes. Therefore, it can be used as a stand-alone rendering program for computer animation production and commercial film industry.

The main contribution of the paper can be summarized as follows. First, we present a fast brush stroke variation method using 2D multitexturing. Second, we introduce a new example-based 1D texture scanning method to obtain realistic texture map from real paintings. Third, we propose a new technique to mimic diffusion effect of black ink painting using 1D multitexturing. By combining these methods, OBIRS can generate realistic black ink images in real-time.

The remainder of the paper is organized as follows. In Section 2, we briefly review the related work. Background knowledge of oriental black ink painting and the basic approach of the system are covered in Section 3. In the following three sections, we explain the three major components of the system: the feature line (Section 4), interior shading (Section 5), and media (Section 6) layers. Experimental results are given in Section 7 and we conclude this paper in Section 8.

2. Related Work

NPR systems developed for oriental black ink painting can be divided into two categories in terms of the user intervention: interactive drawing systems and automatic rendering systems. A drawing system provides an interactive tool with which a user can draw an image in the black ink painting style. In contrast, a rendering system generates a black ink style image from a given 3D object without user intervention. We review the drawing systems first.

Strassmann [7] developed a system which simulates the traditional Japanese art, called *sumi-e*, which is one kind of oriental black ink painting. His work is known as the first NPR research on oriental black ink painting. His brush model consists of 4 components: brush, stroke, dip, and paper. This model is simple but effective for expressing oriental brush effects. Nishita *et al.* [15] presented a modeling technique for a brush trajectory using Bézier curves. The technique has an advantage that the resulting trajectory is very smooth but computing time increases with the number of control points in the Bézier curves. Lee [8, 9] suggested an interactive system for drawing oriental black ink painting. He analyzed each component of black ink painting and proposed the diffusion function of actual black ink. Based on the analysis, he

[1] Blank, the area that remains untouched, is one of the key features in black ink painting. Painters leave some parts untouched or unpainted intentionally for various reasons.

presented models and simulation algorithms for 3D brush, ink, and paper. Way *et al.* [16, 17] presented a method to synthesize rock textures widely used in black ink painting. They showed that nice results can be obtained by texture mapping. Way *et al.* [18] suggested a method of rendering trees which are the essential painting objects in black ink painting. They limit the problem domain to "trees"and demonstrate plausible results. Yu *et al.* [10] presented a model-based approach for simulating oriental black ink painting. They proposed a local equilibrium model to represent the diffusion effect of black ink on a paper. Chan *et al.* [19] suggested two methods to create Chinese painting. The first one is creating Chinese painting animation and the other one is an expressive paint tool. Chu *et al.* [20] presented a physically-based simulation method of ink dispersion in absorbent paper. The dispersion effect of black ink is crucial for black ink painting and results are plausible.

Yeh *et al.* [11] presented an oriental black ink rendering system of 3D meshes. They divided the rendering process into two parts: borderline stroke making and interior shading. A 2D brush model is used to express the brush effects on the borderlines. For interior shading, a Gouraud-shaded image is converted to the black ink painting style using quantization and blurring. Kang *et al.* [12, 13] presented a real-time black ink rendering system with hardware acceleration. They newly suggest a hardware acceleration algorithm for black ink rendering which runs in normal graphics hardware. Grabli *et al.* [21] proposed an image creation method inspired by programmable shaders. In their method, operations are controlled through user-defined procedures. They demonstrated that their method can be applied to the Japanese line drawing style.

The systems described in [11, 12, 13] are similar to our system, OBIRS, in that the goal is to render 3D models in the black ink painting style. However, the resulting images from the systems do not properly reflect the characteristics of black ink painting. We analyze the characteristics of black ink painting to generate realistic results. As a result, our system can produce black ink rendering effects comparable to real paintings.

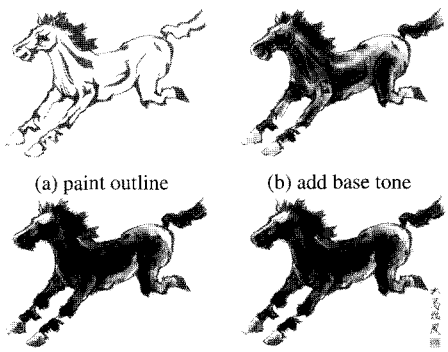
3. Basic Approach

Black ink painting is one kind of oriental painting styles. The painting style is originated from ancient China. That is why oriental black ink painting is also known as Chinese painting in western countries. It became popular in eastern countries, especially in China, Japan, and Korea. In Japan, black ink painting is called "sumi-e". The unique style which uses only light and shade of black ink is known as "soomook" in Korea. These countries have developed black ink painting style in their own ways. Therefore, each country has its own black ink painting style.

Even though each country's black ink painting style is different from others', it has many things in common. To find common factors, we first analyze the drawing steps of black ink painting. As a result, we found that important common factors exist in the drawing steps.

3.1 Drawing steps

The sequence of drawing steps varies according to the painting style and the object type used in black ink painting. However, the following steps are common to most painting styles and object types [6]. Based on our analysis, we can summarize the drawing steps of black ink painting as shown in Fig. 1.



(a) paint outline (b) add base tone

(c) add detail tone (d) sign and seal

그림 1: Drawing steps of black ink painting

1. Paint outline
   The painter sketches feature lines to identify the overall shape. In general, the painter uses a small or extra-small brush for this step. Sometimes, however, the painter uses a medium or large brush to sketch feature line in one stroke.

2. Add base tone
   After finishing the outline painting, the painter uses light ink and soft-fur brushes to fill in spaces between the outlines. The brush strokes must suggest textures and shapes of the object. The tone in each stroke varies according to some factors, such as distance.

3. Add detail tone
   The painter adds detail features on the parts for which emphasis is desired. Since the painter already drew feature outlines in the first step, he or she adds details which are missing. These missing details almost occur in the interior of the object. In the figure, the painter adds details in some body parts and fronts legs.

4. Paint background
   After finishing the main objects, the painter draws the background if necessary (In Fig. 1, we omit this step.). Basically background objects are drawn roughly compared with foreground objects. In most case, therefore, the painter uses large brush to draw background.

5. Sign and seal
   Finally, the painter signs the painting with intense ink and stamps the seal. The signature and stamp appears in the lower right corner of the painting in most cases. But location can vary for the balancing of the painting.

3.2 System overview

The first drawing step of black ink painting is for drawing feature lines of an object. The second and third steps have the same functionality of drawing interiors of an object. However, any special handing is not needed for the fourth drawing step, because we can render background objects in a similar way. Even though the fifth drawing step is crucial for black ink paining, we will omit this step because calligraphy is beyond our interest. According to this analysis, we design OBIRS to consist of two layers: feature line layer and interior shading layer, which are combined to generate black ink style images from 3D meshes. The feature line layer detects and renders the outlines of an object to convey the prominent features, as in Fig. 2(b). The interior shading layer corresponds to the steps in Fig. 2(c) and fills the interior of the model to produce the drawing effects represented in object interiors. However, in Fig. 1, the paper effect is not contained, which is an important characteristic of black ink painting. The result of black ink painting can differ according to the types of paper used. Hence, we include one more layer, the *media* layer, in OBIRS to provide the paper effect as shown in Figs. 2(d). As a result, OBIRS consists of three layers: feature line, interior shading, and media layers.

The framework of OBIRS is based on the drawing steps of black ink painting. We found that, however, our framework is very similar to that of many existing NPR systems, for example, WYSIWYC NPR[22]. Finally, we can conclude that the general framework widely used by existing NPR systems can be adapted to the black ink painting style.

3.3 Characteristics of black ink painting

According to our survey, we found that the framework of OBIRS is similar to existing NPR systems. On the other hand, the characteristics of black ink painting differentiates it from other painting styles. In this subsection, therefore, we describe the characteristics of black ink painting in detail. Also we mention why they are important and how we implement them in our framework. Black ink painting has characteristics which can be summarized as follows [6].

- Brush effect
  Connected long brush strokes are an important feature of black ink painting. The variable width in a stroke represents stress or emphasis. Detailed explanations will be given in Section 4.

- Diffusion effect
  Diffusion is an important feature of this artistic style. It is controlled by the amount of water and pigment in ink. A kind of halo in the boundary of a stroke adds profound impression to the style. Simulating diffusion effect is very time-consuming, so it is not suitable for real-time application. We present a new approach to mimic diffusion effect in Section 5.3.

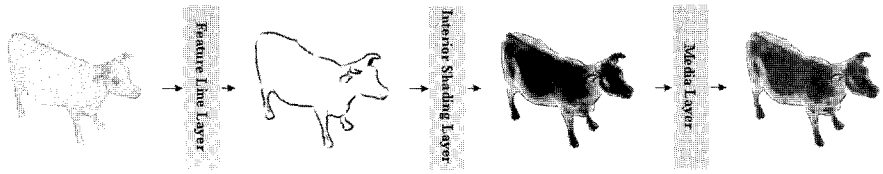
- Blank is beautiful
  Blank is not a vacant space, rather it represents feeling of freedom. It plays an important role in black ink painting. And it makes black ink painting differ from other Western paintings. Blank effect is implemented using the bright area of 1D texture map.

- Conceptual painting
  Objects are modified by transformations, such as simplification, abbreviation, emphasis, and/or exaggeration, according to the painter's aesthetic. Painters draw their paintings using a minimal number of brush strokes to express their aesthetic feelings. To meet the needs of conceptual painting, we connect nearby small brush strokes to form a long connected stroke in feature line layer.

- Planar and representative expression
  Perspective and shadows are ignored, while the main focus is on the existence of objects. This aspect differentiates black ink paintings from Western paintings in which perspective and shadows play important roles.

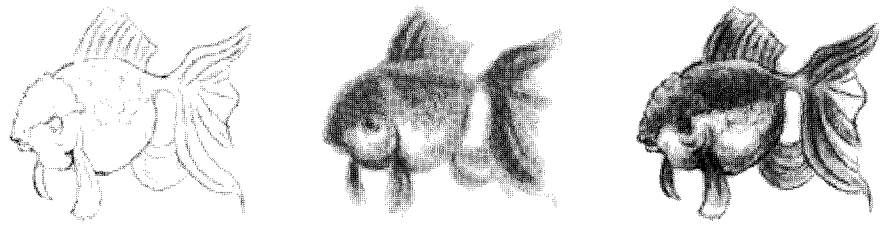
3.4 Painting styles

In black ink painting, there are various painting styles according to painting objects [6]. However, we already mentioned that two factors are in common among various painting styles in Section 3.1: feature line drawing for painting important outlines and interior drawing for painting object's interiors. Our analysis corresponds to the most popular painting styles (as shown in Fig. 3) of black ink painting. In this paper, we will focus on these three styles. We



(a) input 3D model (b) feature line rendered (c) interior shaded (d) paper effect added

그림 2: Rendering steps of OBIRS



(a) Outline style (b) Spontaneous style (c) Outline-spontaneous style

그림 3: Painting styles of black ink painting

will not mention other styles because describing all the styles and techniques of black ink painting in detail is beyond our scope.

- Outline style (Fig. 3 (a))
  This style depends only on smooth outlines without any painting.

- Spontaneous style (Fig. 3 (b))
  In this style, silhouettes are ignored and object shapes are represented by brush strokes.

- Outline-spontaneous style (Fig. 3 (c))
  This style is the combination of the outline and spontaneous styles.

We can obtain images with different styles of black ink painting by selectively using the feature line and interior shading layers of OBIRS. First, outline style images can be generated if we use only the feature line layer without the interior shading layer. In this case, only feature lines are drawn and the object interior remains untouched, e.g., as shown in Fig. 3(a). Second, we can obtain spontaneous style images by taking the interior shading layer only. This style ignores feature lines and focuses only on interior shading. The bamboos in Fig. 16(b) are rendered in this style. Third, we can generate the hybrid *outline-spontaneous* style images by using both feature line and interior shading layers. This style combines the effects of both outline and spontaneous styles.

4. Feature Line Layer

In this layer, feature outlines of the input 3D model are detected and rendered. There are various types of features in NPR: silhouettes, creases, boundaries, and so on. Since feature detection techniques are common in most NPR algorithms, much research has been done so far. Refer to [1, 2, 23, 24, 25, 26, 27] for the details of feature detection. Recently, DeCarlo et al. [28] introduce a powerful algorithm known as *suggestive contours* to detect features. Suggestive contours convey an object's shape precisely and they give additional visual cues compared with silhouettes, creases, or boundaries. Therefore, we decide to use suggestive contours as the feature lines in OBIRS.

After detection, we perform visibility test using ID reference image to extract only visible parts of detected features [29, 25]. For stylization, long smooth brush paths must be obtained by linking individual short line segments [25, 26]. Finally, brush paths are stylized using 2D texture mapping in OBIRS. We construct rib vectors on brush paths for texture mapping.

There are various simulation-based methods developed to imitate the brush effects of real black ink paintings [8, 19, 20]. These simulation techniques can produce impressive results when they are used for rendering detected feature lines. However, the techniques need much computation and do not provide fast rendering speed required in OBIRS. Therefore, we use 2D texture mapping for feature line rendering to generate impressive black ink rendering results at interactive time. Even though texture mapping runs

fast, the quality and effect of resulting images varies according to the quality of texture maps used. The most widely used way to obtain texture maps is to scan from real brush stroke images. This approach is very intuitive and generates artistic results. Fig. 4 demonstrates that the resulting images are comparable to real black ink paintings. Fig. 4 also shows that various brush effects can be obtained by using different brush textures.

Nonetheless, texture mapping approach has some limitations. First of all, we need to scan a large number of texture maps to express various effects of brush strokes. Second, it is difficult to change the style of scanned texture maps in a subtle way, where the subtle change of brush strokes is one of the characteristics of black ink painting. In the remainder of this section, we explain new techniques to generate more realistic black ink rendering results using texture mapping which resolve the limitations of scanning approach.

4.1 Brush effect variation

Even though artistic results can be obtained using scanned texture maps, simply scanning all of the necessary brush textures would take much time and incur storage overhead. To resolve the problem, we propose a texture synthesis method which generates several variations from a given prototype texture by applying image processing operations, such as intensity change and contrast control.

We chose intensity and contrast as basic operations in our implementation, because these operations coincide with the effect of water and pigment of black ink. The more water in a brush, the brighter the brush stroke becomes. On the other hand, the less water in a brush, the darker the brush stroke becomes. We can obtain rough and broken brush strokes with more pigment while smoother brush strokes with less pigment. Fig. 5 shows an example. From a prototype texture, the texture of a dry brush can be generated by controlling the contrast. The textures for bright and dark brushes can be obtained by changing the intensity.

We want to set intensity control parameter from -1.0 to 1.0 where -1.0 means the darkest and 1.0 means the most brightest. Increasing intensity results in shifting the histogram towards white. Likewise, decreasing intensity results in shifting the histogram towards black. As a result, the intensity of resulting texture map is determined by

$$I_{out} = I_{in} + \alpha_{intensity}, \tag{1}$$

where -1.0 $\leq \alpha_{intensity} \leq$ 1.0. After computation is done, I_{out} value must be clamped to the value between 0 and 1.

And we want set contrast parameter from -1.0 to 1.0 where -1.0 means the smoothest and 1.0 means the sharpest. We can adjust contrast stretching out or compress the histogram of the input texture map. If we stretch the histogram out, we get the result with contrast increased, while compressing histogram generates the result with contrast decreased. The contrast adjustment can be determined by

$$I_{out} = (I_{in} - 0.5)(1 + \alpha_{contrast}) + 0.5, \tag{2}$$

where -1.0 $\leq \alpha_{contrast} \leq$ 1.0. After computation is done, I_{out} value must be clamped to the value between 0 and 1.

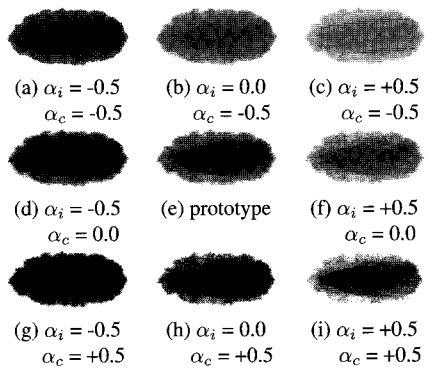


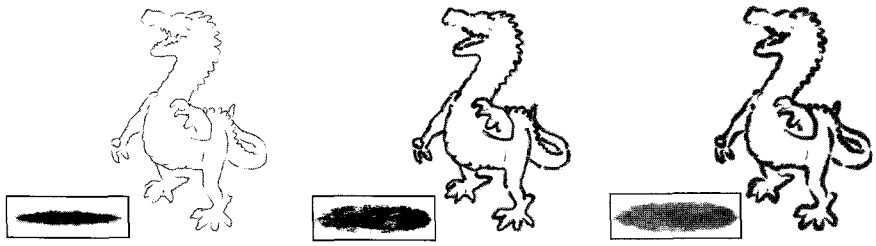
그림 5: Stroke synthesis examples according to $\alpha_{intensity}$ and $\alpha_{contrast}$ from a prototype texture in figure (e). (α_i and α_c are abbreviations of $\alpha_{intensity}$ and $\alpha_{contrast}$ respectively)

Increasing the intensity and the contrast too much will make light pixels pile up at pure white. It destroys highlight details. Likewise, decreasing the intensity and the contrast too much destroys shadow details. Empirically, we find that a value between -0.7 and 0.7 is good for $\alpha_{intensity}$ and $\alpha_{contrast}$.

By synthesizing brush textures from scanned textures on the fly, we do not have to scan all the textures and we can also avoid storage overhead. Furthermore, the computational overhead for image processing operations is negligible. Therefore, we can generate various result images with little computation overhead.

4.2 Brush style variation

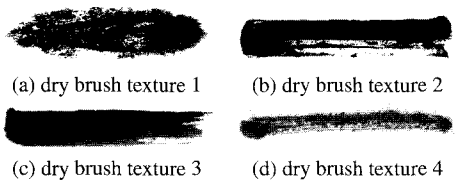
In the previous subsection, we demonstrate that we can generate realistic results using texture variation method. This method is efficient in changing intensity and contrast of scanned brush strokes. However, it is impossible to change the style or the shape of scanned brush textures, as shown in Fig. 6. For a dry brush stroke, a lot of variations exist. This variation comes mainly from unpainted dots and regions. The style variation of brush stroke is very popular in black ink painting. This effect makes resulting images so realistic, but we cannot achieve this effect except scan-



(a) dark brush (b) dry brush (c) wet brush

그림 4: Feature line rendering examples: The lower left rectangles show the texture images used.

ning from real brush strokes. Scanning all necessary brush strokes is time-consuming and results in storage overhead. Therefore, we propose an efficient method to resolve the problem.



(a) dry brush texture 1 (b) dry brush texture 2

(c) dry brush texture 3 (d) dry brush texture 4

그림 6: Examples of stroke textures captured from real paintings.

To guarantee real-time performance while generating realistic effects, we propose a new method based on multitexturing capability of OpenGL. The basic idea is simple: Express the effects as mask textures and blend input brush textures with these mask textures.

Fig. 7(a) shows an input brush texture. Fig. 7 (b) and Fig. 7 (c) are mask textures which will be applied to the input brush texture. By blending mask texture in Fig. 7 (b) with the input texture in Fig. 7 (a), we can synthesize dry brush stroke texture as shown in Fig. 7 (d). Also brush stroke texture in Fig. 7 (e) can be obtained by applying mask texture in Fig. 7 (c). We can generate brush stroke texture in Fig. 7 (f) by applying both mask textures. Fig. 8 demonstrates that our approach generates realistic results very fast using multitexturing capability.

Computational overhead on multitexturing exists because each texture mapping operation is performed sequentially. However, this computational overhead is not so crucial because texture mapping operation itself is implemented with hardware acceleration. This method is efficient since we can reuse generated mask textures. Also a number of mask textures can be applied in layer until plausible results are generated. Once we generate some mask textures, many brush stroke textures with various effects can be obtained by applying a combination of existing mask textures. Detailed ex-

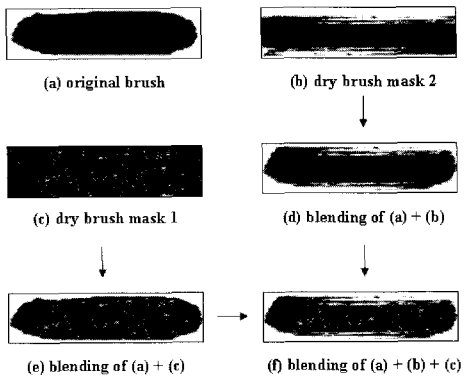


그림 7: Brush texture blending examples: We can synthesize various effects of stroke using multitexturing technique.

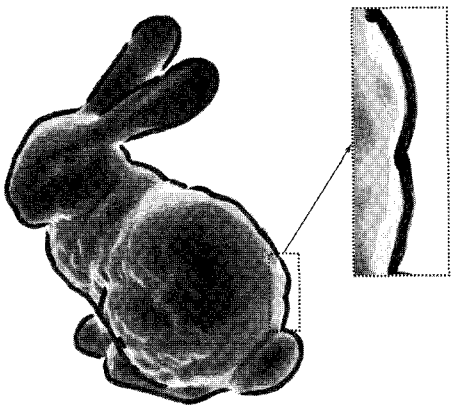


그림 8: Multitexturing example: Upper right rectangle shows zoom-in of brush strokes generated by multitexturing.

planation and implementation details of multitexturing of OpenGL can be found in [30].

4.3 Width control of brush strokes

When a painter draws black ink paintings, width of brush strokes varies according to criteria as shown in Fig. 9. According to our survey, the main factors which determine width are the distance from the viewer and the curvature [6]. The front-left leg is painted more thicker than the front-right leg because the front-left leg is located closer than the front-left leg. The tiptoe part of the front-left leg is painted wider than its other parts because the curvature of tiptoe is higher.

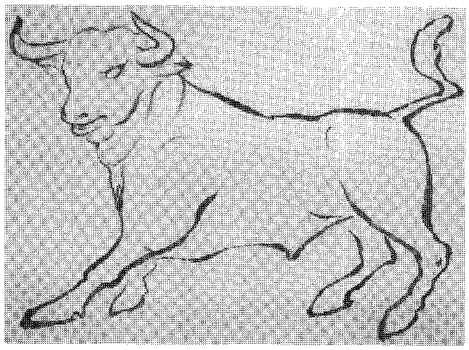


그림 9: Width control in real painting example. Courtesy of Soyoung Lee

Generally, when we draw in the canvas, we assume that, the closer a part is to the viewer, the more important the part is, and vice versa. So if a stroke is close to the viewpoint, the stroke must be thickly drawn. And if a stroke is far from the viewpoint, the stroke must be thinly drawn. To control the width according to the distance from the viewer, we can define the $distancefactor$, $\kappa_{distance}$ as the the reciprocal of the distance from the viewer to the center of a rib vector. In our implementation, we re-map $\kappa_{distance}$ value between 0 and 2: "0" means that the center of a rib vector exists in the rear clipping plane and "2" means in the front clipping plane. The width of a rib vector according to $distancefactor$ can be determined by

$$Width_{distance} = Width_{initial} \times \kappa_{distance}, \quad (3)$$

where $0 \leq \kappa_{distance} \leq 2$.

However, too small or too large values will result in overlapping artifacts of texture mapping coordinates. In our experiments, we find that the distance factor between 0.5 and 1.5 is good: 0.5 means the farthest and 1.5 means the closest.

Painters widen the texture mapped region at the parts where the curvature values are large. Similarly, the region is made narrow if the curvature value is small. For this effect, we define $curvaturefactor$, $\kappa_{curvature}$ as the curvature in 2D space. In our implementation, we measure the $curvaturefactor$ as the dihedral angle formed by the center of a rib vector with two center points of its two adjacent rib vectors. Rib vectors are generated along the angle bisector of a brush stroke. Using rib vectors, we can present breadth to the stroke. Refer to [25] for more details. The width of a rib vector according to $curvaturefactor$ can be determined by

$$Width_{curvature} = Width_{initial} \times \kappa_{curvature}, \quad (4)$$

where $0 \leq \kappa_{curvature} \leq 2$.

Just like the distance factor, if we use too small or too large values, swallow tail artifacts may occur. Empirically, the curvature factor between 0.5 and 1.5 generates good results: 0.5 means flat area and 1.5 means curved area.

Finally, the width of a rib vector is determined by multiplying $distancefactor$ and $curvaturefactor$ with the the given width of a rib vector.

$$Width_{result} = Width_{initial} \times \kappa_{distance} \times \kappa_{curvature}, \quad (5)$$

In other words, the width of the region used for texture mapping is varied according to the distance from the viewer and the detected feature curvature. This effect reflects that a closer part has a wider width and a brush tends to paint a larger area when it makes a big turn in its movement. As a result, we obtain a resulting image as shown in Fig. 10. This figure shows that the width of strokes is controlled properly.

5. Interior Shading Layer

The purpose of interior shading layer is to imitate the shading effects of object interiors. In black ink painting, a large- or medium-size brush is used with light ink for interior shading. Sometimes painters draw a wide stroke using the side of a brush to paint a larger area at a time. In an extreme case, painters even roll brushes to fill the interior [6]. As a result, even though brushes are used, the brush effect is often not noticeable in interior shading.

Based on our survey on black ink painting, we present some requirements for interior shading. First, computation speed must be fast because OBIRS is supposed to be a real-time system. Second, the interior of an object has a smooth transition between three or four tones [6]. To meet the requirements, we choose 1D tex-

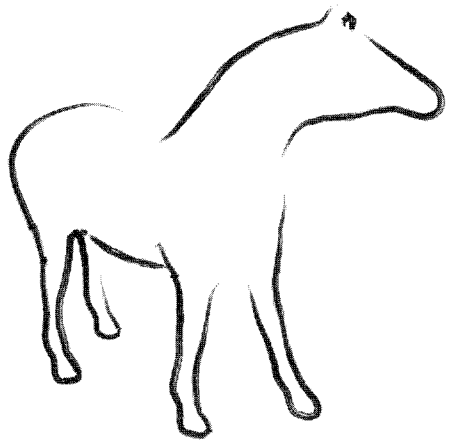


그림 10: Brush stroke width control example: The width of brush stroke is controlled by the distance and the curvature.

ture mapping as the basic approach for the interior shading layer. First of all, 1D texture mapping is fast because it is supported by hardware acceleration. Moreover, smooth transition can be easily implemented using 1D texture mapping.

1D texture mapping is to map a 1D texture map to a surface. And a 1D texture map is an one-dimensional array of colors and scalar texture coordinates are used for mapping the texture onto an object surface. 1D texture maps used for texture mapping mainly affect the quality of results. To obtain natural and realistic resulting images, the following factors should be taken into consideration for interior shading using 1D texture mapping.

The first one is how we can obtain 1D texture maps with gradually changing tone. One possible method for obtaining texture maps is to generate texture maps using commercial softwares, such as the gradation tool of Adobe PhotoShop. In cartoon rendering, 1D texture maps have strict boundaries between different colors or intensities to produce hard edges on the object surface [31]. In contrast, in black ink rendering, we design 1D texture maps to contain smooth transition between intensities to generate smooth tone changes in object interiors. The amount of the blank and the global tone of rendered images can be easily controlled by properly arranging the gray levels in the texture maps. And the texture map should contain three or four levels of intensities because usually three or four tones are used in black ink painting [6]. Unfortunately, the results using texture maps generated in this way look monotonous. To resolve the problem, we need to scan texture maps from real paintings. However, unlike a 2D texture map scanned for a brush stroke, extracting a good 1D texture map from a real painting is difficult because it is not obvious from which line or curve we have to sample the texture map. In Section 5.1, we introduce a new example-based 1D texture scanning method.

Second, texture maps should contain abrupt changes to represent the natural effect of a real painter. This natural effect can enhance the results of interior shading. In Section 5.2, we present a method to enhance natural effect by adding randomness.

Third, diffusion effect is very important in interior shading even though brush effect is not noticeable. As we mentioned in Section 2, much research has been done in simulating diffusion effect in black ink painting. These methods are based on physical simulation approach, so they are time-consuming. To meet the goal of real-time system, we present a new technique which mimics diffusion effect described in Section 5.3.

5.1 Example-based 1D texture scanning

If we can capture 1D textures from real paintings, interior shading results are expected to be more realistic and natural. However, it is not obvious how we can scan 1D textures from real paintings. In this paper, we introduce an example-based 1D texture scanning method which maximizes the naturalness of interior shading. Basically, 1D texture scanning is assigning intensity values to their corresponding texture coordinates. But there are multiple intensity values which correspond to one texture coordinate. We assign the average of multiple intensity values, as a result, a scanned 1D texture map can be obtained.

We will explain the process of our example-based scanning approach in detail as shown in Fig. 11. A painter draws a black ink painting (Fig. 11 (a)) after looking at the 3D model (Fig. 11 (b)). The viewpoint and shape of the model should be as similar as possible to get a correct 1D texture. In the painting, feature strokes are rendered which are unnecessary things for our purpose. Therefore, we delete the feature strokes and unnecessary background manually (Fig. 11 (c)). If the painter draws a 3D model in spontaneous style, we can skip this step. In our system, we calculate a scalar value for each vertex using the dot product of the view direction and vertex normal. And we use the assigned scalar values at vertices as the texture coordinates. So just rendering input 3D model with these values generates a texture coordinate map (Fig. 11 (d)). Finally, we can compare these two images pixel by pixel to generate resulting 1D texture (Fig. 11 (e)). Interior area gets darker and darker if it becomes far from silhouette regions using this approach. Fig. 11 (f) shows a rendering result with the scanned 1D texture map. The result is more natural and realistic. We conclude that scanned 1D textures guarantee realistic and natural interior shading results.

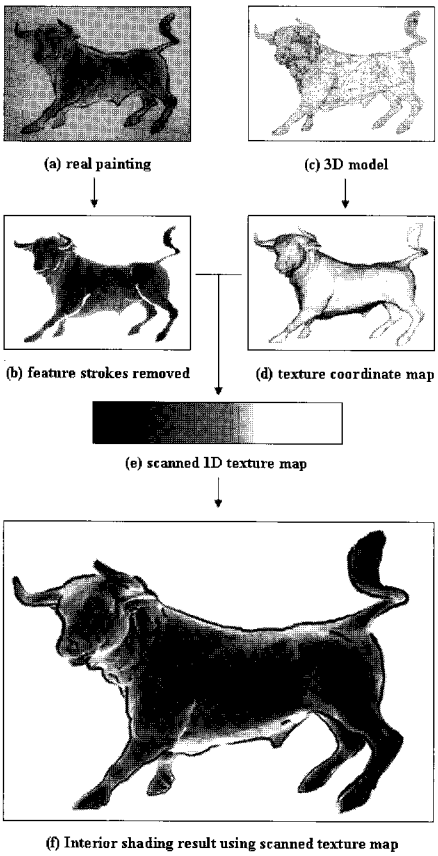


그림 11: Process of 1D texture scanning

5.2 Enhancement of natural effect

When a painter draws a painting, there must be abrupt changes in interior shading. A painting looks artificial without any abrupt changes. Our example-based 1D texture scanning can generate realistic results. The strong point of this technique is to add "natural effect" in interior shading. However, if an input mesh is very smooth, the effect will decrease as shown in Fig. 12 (a). To resolve the problem, we propose a new approach to enhance the natural effect.

A rendering result looks unnatural if interior shading is monotonous as shown in Fig. 12 (a). Our basic idea to enhance natural effect is adding "randomness" to the object's interior. However, we have some candidates to apply randomness in interior shading. First, we can apply randomness in texture coordinates at each frame. However, this approach cannot guarantee any temporal coherence between frames. Therefore, we chose vertex normal as our candidate. If we add randomness to vertex normal, we do not have to worry about temporal coherence. The result shows that the randomness enhance the naturalness of the result as shown in Fig. 12 (b).

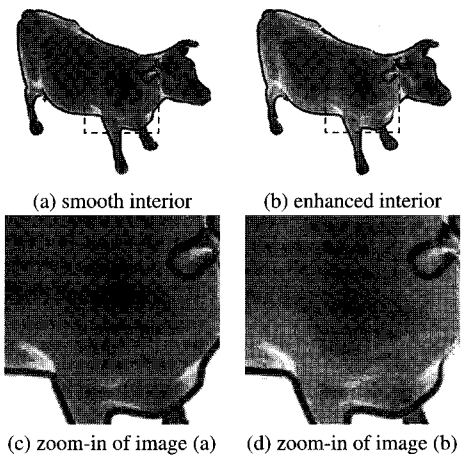


그림 12: Enhancement of interior by adding randomness to vertex normals

5.3 Emulation of diffusion effect

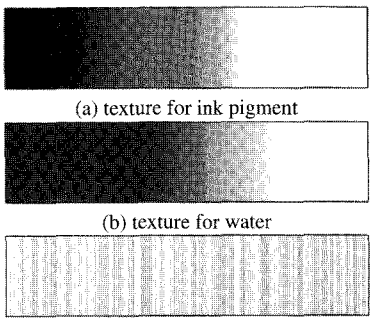
In the previous subsection, we explained the method to enhance the interior shading effect using only one 1D texture map. By adding randomness effect, we can generate more realistic results. However, it is known that simulating the diffusion effect of black ink using 1D texture mapping is almost impossible [9]. Despite of this limitation, we chose 1D texture mapping for real-time perfor-

mance. Even though it is difficult to simulate exact diffusion effect, we want to mimic diffusion effect as much as possible, because diffusion effect is one of the most important characteristics of black ink paining.

In physics, the term "diffusion" is defined as the random movement of particles towards less concentrated regions [8, 9]. In reality, ink diffusion is a complex interaction between ink pigment, water, and paper. Therefore, simulation methods are needed to obtain good results with diffusion effect. In our implementation, however, we confine diffusion effect to an interaction between ink pigment and water. A diffusion result looks remarkable as the amount of water increases while the amount of pigment decreases. On the contrary, diffusion effect is negligible when the amount of ink pigment dominates the amount of water.

In this paper, we propose a new approach to emulate the diffusion effect using 1D texture mapping. The basic idea is to increase or decrease the amount of water while the amount of pigment is fixed. As the amount of water increases, ink pigments may travel farther and it becomes more lighter in intensity. If the amount of water decreases, ink pigments will stay around and it becomes more darker in intensity. To represent the amount of pigment and water, we use texture maps as shown in Fig. 13 (a) and Fig. 13 (b) respectively. These textures may be smooth. However, textures with randomness generate better results because the distribution of pigment is not uniform. Therefore, we use the texture map obtained using example-based 1D texture scanning as the texture for pigment in the example. Then, two textures are blended using multitexturing. Since we use smooth texture for water, it smoothes out the result. In effect, the amount of pigments' movement differs according to the amount of water. Unfortunately, it is impossible to implement this effect using 1D texture mapping. To overcome the limitation, we use one more texture to represent the amount of pigments' movement. The amount of pigments' movement is correlated with the amount of water. In practice, however, drastic changes in 1D texture maps result in artifacts, such as Mach band effect. We found that randomness as shown in Fig. 13 (c) generates plausible results without suffering from any artifacts. Finally, these three textures are blended using multitexturing in OpenGL.

Fig. 14 shows the result of diffusion effect. By adding diffusion effect (Fig. 14 (b)), the result looks more plausible. You can find that the shading effect gets better in the bright side of the interior. Because our approach just mimics the diffusion effect, however, the quality of results depends on the texture maps used. Also too much randomness factor will degenerate the shading result.

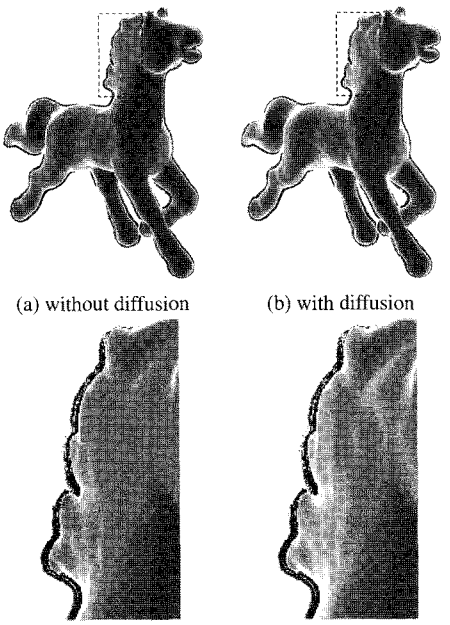


(a) texture for ink pigment

(b) texture for water

(c) texture for randomness

그림 13: Texture maps used for diffusion effect



(a) without diffusion (b) with diffusion

(c) zoom-in of image (a) (d) zoom-in of image (b)

그림 14: Comparison results of diffusion effect

6. Media Layer

The purpose of the media layer is to imitate the paper effect in black ink painting. The paper effect underlying the object image is one of the most important characteristics in black ink painting. The papers used in black ink painting are very rough and textured. Several types of papers are used in black ink painting and the paper type has influence on the feeling of the painting result. In the media layer of OBIRS, we implement the effect of a *rice paper* because it is the most widely used paper type in black ink painting [6].

Simulation techniques were developed to imitate the effects of various types of papers in different painting styles. Curtis *et al.* [3] proposed a simulation model of the paper used in watercolor painting. Lee [9] presented a simulation algorithm for paper effects in oriental black ink painting. Although simulation methods can generate realistic results, they may not be proper for real-time rendering systems due to high computational overhead.

To avoid computational overhead, we represent the roughness of a rice paper using a simple function. In our implementation, we adopt the notion of a *height field* proposed by [3]. On the peak where the height is 1, the pigments are well absorbed, while in the valley with height 0, the pigments are slightly absorbed. To obtain a height field for the rice paper effect, we analyze the scanned image of the paper and repeat the extracted basic pattern with random perturbation. The constructed height field is stored as the α values of the pixels in a screen size image. In the rendering process, the paper effect is realized by applying α-blending to the feature lines and interiors drawn on the image containing the height field. A similar approach to implement paper effects was used in [22].

Fig. 15 shows a zoom-in of the rice paper effect generated by this approach. Since paper effect is implemented by texture mapping, this layer has little computational overhead. However, a limitation of our approach is that the paper effect does not change with viewing parameters because we use a single fixed height map.

그림 15: Zoom-in of the rice paper effect

7. Experimental Results

We have implemented the proposed black ink rendering system for 3D meshes on a Pentium IV 3.6GHz PC with an nVIDIA GeForce 6800 Ultra graphics card. Fig. 16 shows rendering images of various 3D meshes in different painting styles.

Fig. 16(b) and the crane of Fig. 17(b) are examples of the outline style. In Fig. 16(b), a dry brush texture is used for feature lines so that the feather of the bee looks rough. In contrast, a wet brush texture is used to represent the smoothness of the crane's feather in Fig. 17(b). Fig. 17 (a) and the bamboos in Fig. 17(b) are examples of the spontaneous style. We use a bright 1D texture for the bamboos in Fig. 17(b) to express the elegance of bamboos. Figs. 16(a), (c), (d), (e), (f) and the dead tree in Fig. 17(b) are examples of the outline-spontaneous style. The textures used for interior shading of the dead tree and the bamboos in Fig. 17(b) are different because the interior of a dead tree is darker than that of a bamboo in general.

Temporal coherence has become an importance issue in NRP society these days. One of the strong points of our algorithm is that it is easy to generate black ink animation. However, animation results will not be plausible if successive frames are temporally incoherent. To obtain temporally coherent animation, we adopt the algorithm introduced by Kalnins *et al.* [32]. Their algorithm is known to be suitable for stroke-based rendering systems, just like ours.

Table 1 reports the frame rates of the rendering examples shown in this paper. From Table 1, we can see that our system can render 3D meshes with relatively high complexity at interactive time. The models used in this paper contain from several to tens of thousands of polygons. In general, smaller models are used in computer games and virtual environments and in that case, our system can generate black ink style images of a 3D model in real-time.

figure	model	# polygon	FPS
Fig. 2	Cow	8,706	31.3
Fig. 15	Bull	18,504	16.2
Fig. 16 (a)	Octopus	16,522	14.4
Fig. 16 (b)	Bee	37,358	11.2
Fig. 16 (c)	Dragon	53,458	5.9
Fig. 16 (d)	Feline	99,732	2.8

표 1: Frame rates of various models

8. Conclusions

In this paper, we presented a rendering system for 3D meshes which generates images in the oriental black ink painting style. Our system has several good properties. First, the system allows fast generation of resulting images because it does not contain any

simulation of a physical process. Therefore, it can be directly used in real-time applications, such as computer games and virtual environments. Second, due to the use of textures extracted from real paintings, the system achieves artistic effects in the generated images. Finally, the rendering process is fully automatic after the rendering style and a few parameters have been specified by the user. Hence, a black ink animation can easily be generated by the system without much user intervention.

However, the current system also has some limitations. First of all, any hardware acceleration techniques, such as vertex shader and pixel shader, are not used in the system. Even though our system guarantees real-time speed in most cases, proper incorporation of the techniques into the system will speed up the rendering process. Second, to imitate the abstraction performed by a painter in the drawing process, exaggeration or simplification can be applied to the input model before rendering. However, in this case, it would be important to select the features to be exaggerated or preserved in the simplification. Third, among the characteristics of black ink painting explained in in Section 3.3, the characteristic of conceptual painting is not fully taken in consideration in this paper. We implemented the system to render an object using a minimal number of strokes for conceptual painting property. However, modification of objects such as simplification or exaggeration can express painter's aesthetic feelings very well. In the future, therefore, this modification should be implemented for more artistic results.

참고 문헌

[1] B. Gooch and A. Gooch, *Non-Photorealistic Rendering*. A K Peters, 2001.

[2] T. Strothotte and S. Schlechtweg, *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann Publishers, 2002.

[3] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. Salesin, "Computer-generated watercolor," *ACM Computer Graphics (Proc. SIGGRAPH '97)*, pp. 421–430, 1997.

[4] M. C. Sousa and J. W. Buchanan, "Computer-generated graphite pencil rendering of 3D polygonal models," *Computer Graphics Forum (Proc. Eurographics'99)*, pp. 195–208, 1999.

[5] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-time hatching," *ACM Computer Graphics (Proc. SIGGRAPH 2001)*, pp. 581–586, 2001.

[6] L. Q. Zhen, *Chinese Painting Techniques for Exquisite Watercolors*. North Light Books, 2000.

[7] S. Strassmann, "Hairy brushes," *ACM Computer Graphics (Proc. SIGGRAPH '86)*, pp. 225–232, 1986.

[8] J. Lee, "Simulating oriental black-ink painting," *IEEE Computer Graphics and Applications*, vol. 19, no. 3, pp. 74–81, 1999.

[9] ——, "Diffusion rendering of black ink paintings using new paper and ink models," *Computers and Graphics*, vol. 25, no. 3, pp. 295–308, 2001.

[10] Y. J. Yu, D. H. Lee, Y. B. Lee, and H. G. Cho, "Interactive rendering technique for realistic oriental painting," *Proc. Winter School of Computer Graphics '2003*, pp. 538–545, 2003.

[11] J.-W. Yeh and M. Ouhyoung, "Non-photorealistic rendering in Chinese painting of animals," *Journal of System Simulation*, vol. 14, no. 6, pp. 1220–1224, 2002.

[12] S.-J. Kang and C.-H. Kim, "Real-time 3D sumi-e painting," *SIGGRAPH 2003 Skteches and Applications*, 2003.

[13] S.-J. Kang, S.-J. Kim, and C.-H. Kim, "Hardware-accelerated real-time rendering for 3d sumi-e painting," *Lecture Notes in Computer Science*, pp. 599–608, 2003.

[14] A. W. Klein, W. Li, M. Kazhdan, W. T. Correa, A. Finkelstein, and T. A. Funkhouser, "Non-photorealistic virtual environments," *ACM Computer Graphics (Proc. SIGGRAPH 2000)*, pp. 527–534, 2000.

[15] T. Nishita, S. Takita, and E. Nakamae, "A display algorithm of brush strokes using bezier functions," *Proc. Computer Graphics International '93*, pp. 244–257, June 1993.

[16] D.-L. Way and Z.-C. Shih, "The synthesis of rock textures in Chinese landscape painting," *Computer Graphics Forum (Proc. Eurographics 2001)*, pp. 123–131, 2001.

[17] D.-L. Way, C.-W. Hsu, H.-Y. Chiu, and Z.-C. Shin, "Computer-generated chinese painting for landscape and portraits," *Proc. Winter School of Computer Graphics '2001*, pp. 387–394, 2001.

[18] D.-L. Way, Y.-R. Lin, and Z.-C. Shih, "The synthesis of trees in Chinese landscape painting using silhouette and texture strokes," *Proc. Winter School of Computer Graphics '2002*, vol. 10, no. 2, pp. 499–506, 2002.

[19] C. C. Chan, E. Akleman, and J. Chen, "Two methods for creating chinese painting," *Proc. Pacific Graphics 2002*, pp. 403–412, 2002.

[20] N. S.-H. Chu and C.-L. Tai, "Moxi: Real-time ink dispersion in absorbent paper," pp. 504–511, 2005.

[21] S. Grabli, E. Turquin, F. Durand, and F. Sillion, "Programmable style for npr line drawing," *Eurographics Symposium on Rendering*, pp. 33–44, 2004.

[22] R. D. Kalnins, L. Markosian, B. J. Meier, M. A. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes, and A. Finkelstein, "WYSIWYG NPR: Drawing strokes directly on 3D models," *ACM Computer Graphics (Proc. SIGGRAPH 2002)*, pp. 755–762, 2002.

[23] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes, "Real-time nonphotorealistic rendering," *ACM Computer Graphics (Proc. SIGGRAPH '97)*, pp. 415–420, 1997.

[24] A. Hertzmann and D. Zorin, "Illustrating smooth surfaces," *ACM Computer Graphics (Proc. SIGGRAPH 2000)*, pp. 517–526, 2000.

[25] J. Northrup and L. Markosian, "Artistic silhouettes: A hybrid approach," *Non-Photorealistic Animation and Rendering (Proc. NPAR 2000)*, pp. 31–37, 2000.

[26] T. Isenberg, N. Halper, and T. Strothotte, "Stylizing silhouettes at interactive rates: From silhouette edges to silhouette strokes," *Computer Graphics Forum (Proc. Eurographics 2002)*, vol. 21, no. 3, 2002.

[27] T. Isenberg, N. Halper, S. Schlechtweg, and T. Strothotte, "A developer's guide to silhouette algorithms for polygonal models," *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 28–37, 2003.

[28] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Computer Graphics (Proc. SIGGRAPH 2003)*, pp. 848–855, 2003.

[29] M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. Hughes, "Art-based rendering of fur, grass, and trees," *ACM Computer Graphics (Proc. SIGGRAPH '99)*, pp. 433–438, 1999.

[30] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2, Fifth Edition*. Addison Wesley Professional, 2005.

[31] A. Lake, C. Marshall, M. Harris, and M. Blackstein, "Stylized rendering techniques for scalable real-time 3D animation," *Non-Photorealistic Animation and Rendering (Proc. NPAR 2000)*, pp. 13–20, 2000.

[32] R. D. Kalnins, P. L. Davidson, L. Markosian, and A. Finkelstein, "Coherent stylized silhouettes," *ACM Computer Graphics (Proc. SIGGRAPH 2003)*, pp. 856–861, 2003.

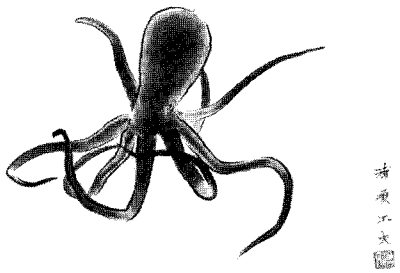
〈저자소개〉

정규만

- 1998년 2월 KAIST 전산학과(학사)
- 2000년 2월 POSTECH 컴퓨터공학과(석사)
- 2007년 2월 POSTECH 컴퓨터공학과(박사)
- 2004년 3월~2005년 3월 미국 미시건대학교 교환연구원
- 2007년 3월~2007년 4월 POSTECH 컴퓨터공학과 박사후 연구원
- 2007년 5월~2009년 2월 삼성전자 정보통신총괄 책임연구원
- 2009년 3월~현재 대구대학교 정보통신공학부 교수
- 〈관심분야〉 비사실적 렌더링, 계산 사진학, 컴퓨터 게임 등

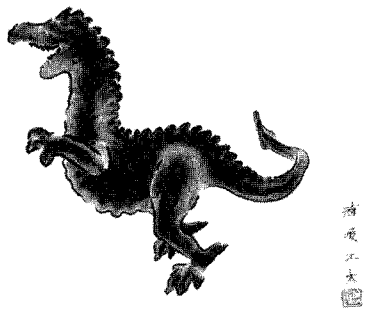
이승용

- 1988년 2월 서울대학교 계산통계학과(학사)
- 1990년 2월 한국과학기술원 전산학과(석사)
- 1995년 2월 한국과학기술원 전산학과(박사)
- 1995년 3월~1996년 9월 미국 City College/CUNY 박사후 연구원
- 2003년 8월~2004년 8월 독일 MPI Informatik 방문 선임연구원
- 1996년 10월~현재 포항공과대학교 컴퓨터공학과 교수
- 〈관심분야〉 메쉬처리, 3D 곡면복원, 비사실적 렌더링, 영상처리 등

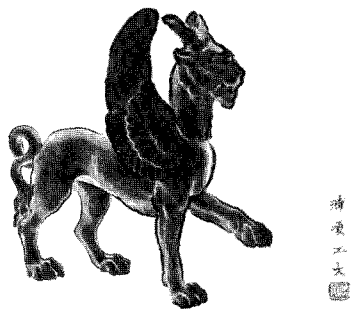


(a) octopus

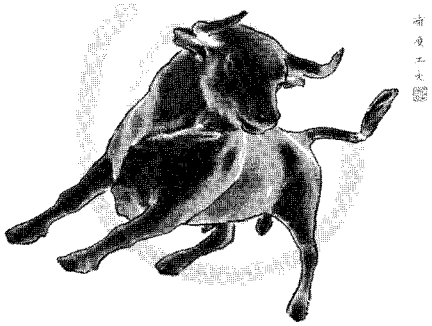
(b) bee



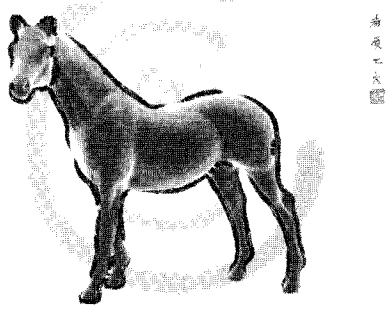
(c) dragon



(d) feline



(e) bull



(f) horse

그림 16: Black ink rendering examples

(a) scene with two dragonflies in a bush

(b) scene with a dead tree, a crane, and a bamboo forest

그림 17: Black ink rendering examples