Artistic Rendering of Function-based Shape Models

by

Shunsuke Suzuki

Faculty of Computer and Information Science
Hosei University
n00k1021@k.hosei.ac.jp

Supervisor: Alexander Pasko

March 2004
Abstract

We explore approaches to artistic (non-photorealistic) rendering of function-based 3D models described in the HyperFun language. Although ray tracing is originally a technique for photorealistic rendering, we dare render HyperFun models in an artistic style using ray tracing. Examples of artistic rendering are watercolor painting, pen or pencil sketch, and “cartoon” style rendering. We tested several techniques and proposed a method for rendering in the pencil sketch style. In this method, we generate the 3D model’s silhouette by analyzing an angle between its normal vector and the view vector. The pencil dots and strokes for interior shading are generated by dithering, random-dithering or straight line segments with the direction obtained by the cross product of its normal vector and the view vector. In interior shading method, we studied which techniques can generate more artistic images. Thus, we obtained synthetic artistic images by utilizing the principle of ray-tracing and different shading techniques.
Chapter 1

Introduction

Realistic rendering is commonly considered a goal of many computer graphics techniques. However, non-photorealistic (or artistic) rendering generating hand-made images is also under intensive research recently. The difference between realistic images and non-photorealistic images is similar to difference between photos and hand-made illustrations. At present, it is quite easy to create realistic images using different computer graphics techniques. On the other hand, if we can produce artistic images automatically, the utility value of computer graphics will become bigger, and computer graphic will become more attractive.

In this work, we consider approaches to artistic rendering of function-based 3D models in HyperFun by utilizing the principle of ray tracing. HyperFun is a 3D function-based modeling language. The models are rendered in the form of

\[ f(x, y, z) \geq 0. \]

When a point \((x, y, z)\) is considered as the point of 3-dimensional space, HyperFun models consist of the set of points observing the rule (the points that 0 and over values are returned). For example, the rule for modeling a sphere having radius 5 and center coordinates \((0,0,0)\) is

\[ f(x, y, z) = 5^2 - (x^2 + y^2 + z^2) \geq 0. \]

The sphere is made from the set of the points that observe the rule. In addition, the point, that \(f(x, y, z)\)’s value is 0, is the model’s surface, and the points, that the value is bigger than 0, is the interior model. It is also possible to make several simple objects observing the rule into the complex model with Constructive Solid Geometry expression. Constructive Solid Geometry represents CSG. CSG is combining simple parts using set operation, and making a complex model.
HyperFun is described in detail in [8]. We also use OpenGL as a tool for implementation of proposed rendering algorithms.

Our software first inputs and compiles a HyperFun model, then it traces rays emitted from each pixel of the image and looks for an intersection point between the ray and the model surface. If a ray-surface intersection point is found, we apply different shading algorithms for generating a non-photorealistic image element at this point. A shading algorithm starts from distinguishing between a model’s silhouette and an inside points of the model projection to render them differently. An image is generated when ray tracing is completed.

A number of approaches have been recently developed for the generation of artistic images of 3D models. An animation method for pen-and-ink illustration was proposed in [3]. The algorithm of [4] represents a non-photorealistic image using “geograffals”. The work [5] approaches the oriental paintings generation. Methods of extracting model’s silhouettes are presented in [1, 6]. The method [2] uses Bezier functions to express pen-strokes. Thus, there is much research, and we consider that the big difference between these researches and our research is to generate non-photorealistic images of function-based 3D models using ray tracing.

We combine the above techniques to generate artistic images of complex function-based models. In the following chapters, the algorithm of extracting model’s silhouette is described. In chapter 3, the interior shading is described, first, using standard dithering, and then we introduce a random-dithering method. Finally, the strokes generation method, which is a feature of the pencil sketch style, is presented. Then, the images of complex models generated using the proposed algorithms are shown and briefly discussed in chapter 4 and processing time is described about a sphere model and a tiger model and conclusions and future work discussion are given in chapter 5, 6.
Chapter 2
Finding Silhouette

To render non-photorealistic images, first of all, we find model’s silhouette. To do this, we use the view vector $v (0,0,-1)$ and the surface normal vector $\nabla f$ at each point of the model’s surface. We assume that the view vector is always perpendicular to the screen. The surface point where the view vector is perpendicular to the normal vector is a silhouette point. In other words, if the cosine value of the angle between the normal vector and the view vector is 0, this point belongs to the silhouette.

$$F(p) = \frac{\nabla f(p)_z}{\sqrt{\nabla f(p)_x^2 + \nabla f(p)_y^2 + \nabla f(p)_z^2}} = 0$$

$F(p)$ is a silhouette point. Please look at the above equation, we only find the point that Z-values of the surface normal vector is 0. This algorithm is shown in Fig. 1. If the cosine value is equal to 0, we consider the point a silhouette and put a black point to the corresponding pixel of the image. However, there is one problem. If only the points with the cosine value equal to 0 are selected, the generated silhouette curve is very thin and invisible the naked eye. Therefore, as a solution, we consider the points with the cosine value between 0 some given threshold (typically 0.2) as silhouette points, and we change the threshold according to the given shape. An image of the extracted silhouette curve of a 3D sphere model is shown in Fig.2.
Figure 1. When we reach a point $p$ of the surface, we check the cosine value of the angle between the surface normal vector $\nabla f$ and the view vector $v$. If the value is 0, the point is a silhouette point.

Figure 2. A 3D sphere model and its extracted silhouette: (a) tested 3D model; (b) extracted silhouette points.
Chapter 3

Interior Shading

The method for rendering interior shade is very important for the non-photorealistic image generation. We test three possible interior shading techniques:

1. dithering
2. random dithering
3. generating strokes.

Dithering and random-dithering are techniques for pen or pencil dots style, and generating strokes is techniques for pencil sketch style. We describe them.

3.1. Dithering

Dithering is usually used for halftone image generation on black/white output devices like dot matrix printers. Here, we apply this technique for interior shading with pencil or pen dots. To apply dithering, we set the 64 values (from 0 to 63) to each elements of an $8 \times 8$ matrix without duplication. This matrix is called a “dither matrix”. The formula for generating the $n \times n$ dither matrix is

$$D^{(n)} = \begin{bmatrix}
4D^{(n/2)} + D^{(2)}_{00}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{01}U^{(n/2)} \\
4D^{(n/2)} + D^{(2)}_{10}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{11}U^{(n/2)}
\end{bmatrix}.$$
\(U^{(n)}\) defined as an \(n \times n\) matrix of 1s, that is,

\[
U^{(n)} = \begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & 1 & 1 & \cdots & 1 \\
1 & 1 & 1 & \cdots & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & \cdots & 1
\end{bmatrix}.
\]

Furthermore, \(D^{(2)}\) is

\[
D^{(2)} = \begin{bmatrix}
0 & 2 \\
3 & 1
\end{bmatrix}.
\]

Consequently, \(8 \times 8\) dither matrix \((D^{(8)})\) generated from the formula is

\[
D^{(8)} = \begin{bmatrix}
63 & 31 & 55 & 23 & 61 & 29 & 53 & 21 \\
15 & 47 & 7 & 39 & 13 & 45 & 5 & 37 \\
51 & 19 & 59 & 27 & 49 & 17 & 57 & 25 \\
3 & 35 & 11 & 43 & 1 & 33 & 9 & 41 \\
60 & 28 & 52 & 20 & 62 & 30 & 54 & 22 \\
12 & 44 & 4 & 36 & 14 & 46 & 6 & 38 \\
48 & 16 & 56 & 24 & 50 & 18 & 58 & 26 \\
0 & 32 & 8 & 40 & 2 & 34 & 10 & 42
\end{bmatrix}.
\]

[7] describes how to make dither matrix in detail. Then, the ray-tracing is applied to a central point of each \(8 \times 8\) “mega-pixel”. For the ray-surface intersection point in the central point of the mega-pixel, we calculate the angle between the normal vector \(\nabla f\) and the light source vector \(l\), and the cosine value \(E\) of the angle. For the \(E\) value changing from 0 to 1, an intensity in the mega-pixel will be set to 64E (changing from 0 to 64). We compare each element of a mega-pixel with each element of the dither matrix, and put a black point to the pixel, where the element of the dither matrix is bigger than the intensity. If not, we do nothing to the pixel. As an example, we introduce this algorithm in Fig. 3.
Figure 3. The case with $4 \times 4$ dithering. When the intensity is 8, black points are put to seven pixels where the element of dither matrix is bigger than the intensity.

The dithering effect can be explained such that the number of rays emitted from the light source and hitting the surface is proportional to the cosine of the angle between the normal vector and light source vector. If the direction to the light source is perpendicular to the normal, the cosine becomes small and there is a little number of rays hitting the surface. If not, the cosine becomes large and there is much number of the rays. Therefore, for mega-pixels with low intensity, many black points are put to the 64 pixels. In addition, we applied a $4 \times 4$ matrix for dithering, but could not get a good result, because of the limited number of intensity levels. $4 \times 4$ dithering only has 16 intensity levels. On the other hand, $8 \times 8$ dithering has 64 intensity levels. An image generated using $8 \times 8$ dithering and $4 \times 4$ dithering is shown in Fig.4. The main problem with dithering is the regular shading patterns, visible in Fig. 4, which are not acceptable for artistic drawing simulation. In $4 \times 4$ dithering, the more regular shading patterns appear than $8 \times 8$ dithering.
Figure 4. An $600 \times 600$ image of a sphere model with $8 \times 8$ dithering. Then, main light source position is set to XYZ-coordinates is $(60,60,100)$ and sub light source position is $(−60,−60,50)$ with Z-axis pointed to the viewer: (a) the entire image with $8 \times 8$ dithering; (b) enlarged part of the image and (c) the entire image with $4 \times 4$ dithering; (d) enlarged part of the image.
3.2. Random Dithering

The difference is that we change the fixed dither matrix to the variable matrix. Whenever we apply dithering, each element of the dither matrix is randomly changed with a prescribed distribution. We assume a variation range of each element of the dither matrix and the uniform distribution of the values inside the variation range. Then, we compare each element of a mega-pixel with each element of dither matrix by using same procedure as $8 \times 8$ dithering. We tested various values of the variation range, could obtain best artistic images, when the value is 10 or 5. The pseudocode of the random dithering with the dither matrix $D$ and the variation range $\Delta$ is as follows:

For each pixel of a mega-pixel excepting silhouette parts and background parts

If

\[
D[I] + \Delta \times \left( \frac{1 - 2 \times \text{rand}(\cdot)}{1.0 + \text{RAND_MAX}} \right)
\]

is bigger than intensity

Put a black point on the pixel

Else

Don’t put a black point on the pixel.

An image generated using random dithering is shown in Fig. 5. As one can observe, the effect of regular shading patterns is significantly reduced, if compared with Fig. 4. Therefore, we can express more artistic shade for pen or pencil dots.
Figure 5. A sphere model image generated using $8 \times 8$ random dithering: (a) the entire image with the value of 10 of the variation range; (b) enlarged part of the image and (c) the entire image with the value of 20 of the variation range; (d) enlarged part of the image.

3.3. Generating strokes for interior shading

To render shapes in the pencil sketch style, it is very important to generate strokes for interior shading. First, we must detect the direction of a stroke. We use the method proposed in [1] to calculate the direction of strokes. The stroke
direction vector \( s \) is calculated by a cross product of the view vector \( \mathbf{v} \) \((0,0,-1)\) and the normal vector \( \nabla f \) at each point of the surface:

\[ s = \mathbf{v} \times \nabla f \]

This direction is tangent to the curve of constant illumination (isophote). We generate strokes for \( 8 \times 8 \) mega-pixel, because we want to represent the intensity by the number of parallel strokes instead of dithering. To get the stroke direction vector, we calculate the cross product of the view vector and the normal vector at each central point of a mega-pixel. Then, we need to generate a straight line segment image from the point \((x,y)\) to the point \((x + s_x, y + s_y)\) going in the stroke direction. We can put a black point to any pixel, which is located on the line. Here, there is a problem. Since the coordinate values of each pixel are integer values, there is almost no pixel exactly located on the line. Therefore, an approximation is used in order to draw a stroke. If the directional distance between the line and the pixel point is less than 0.5, a black point is put to the pixel. This is the basic stroke generation procedure. In addition, since view vector always is \((0,0,-1)\), the Z-coordinate value of the stroke direction vector is always 0. Therefore, we consider only with XY-coordinates value.

We also have to decide the number of the strokes, which should be drawn in each mega-pixel. Then, we utilized effectively the “dither matrix” used before. The number of strokes is determined by the mega-pixel’s intensity or the number of the pixel which a black point is put to. In addition, when two or more strokes exist in one mega-pixel, in order to prevent a stroke overlapping, the strokes are placed by applying parallel translation in the X and Y direction at random distance from the basic stroke. Furthermore, the more the intensity is bigger, the more stroke color is pale. The rule for determining the number of strokes drawn in each mega-pixel is shown in Table 1. This algorithm is shown in Fig.6 and an image with interior shading using strokes is shown in Fig.7.
Table 1. The rule for determining the number of strokes drawn in each mega-pixel.

<table>
<thead>
<tr>
<th>Black point</th>
<th>Intensity</th>
<th>Strokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-9</td>
<td>64-55</td>
<td>0</td>
</tr>
<tr>
<td>10-32</td>
<td>54-32</td>
<td>1</td>
</tr>
<tr>
<td>33-48</td>
<td>31-16</td>
<td>2</td>
</tr>
<tr>
<td>49-64</td>
<td>15-0</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 6. The algorithm of interior shading by strokes.

Figure 7. Rendering of a sphere model using interior shading with strokes: (a) the entire image; (b) enlarged part of the image.
Chapter 4

Experiments

In this chapter, we show and discuss the images of complex constructive shape models in HyperFun (Fig. 8). We could render all models using silhouette detection and strokes. However, small dots appear in the upper part of a chair image and there is also the part that is all black. The main problem of all images is that the width of silhouettes is loose. Especially, the silhouette rendering problem appears at the flat parts and on slightly curved parts of models. The reason is that silhouettes are extracted only according to the normal vector direction and other shape characteristics of the object are not taken into consideration. Usually, the threshold of the cosine value of the angle between each normal vector and the view vector is set up greatly in order to extract all silhouettes, and so there are thin line and thick line. In (a), (b) and (d) images, the range for the cosine is set from 0 to 0.3. For the (c) image, the range is set from 0 to 0.4. However, the range is set from 0 to 0.2 for a simple sphere.
Figure 8. Rendering complex models in the pencil sketch style: (a) chair, (b) ant, (c) tiger and (d) sample HyperFun model.
Chapter 5

Processing Time

In this chapter, we inspect processing time of our rendering about a sphere model and a tiger model. Most of the processing time is to find ray-surface intersection points. An element giving great influence in the time is the Lipschitz value.

5.1. Lipschitz value

The Lipschitz value is greatly related to ray casting step. If the Lipschitz value is big, ray casting step will be small. That is, ray-surface intersection point is found precisely, and the precision of an image is better. However, the processing time will be long.

5.2. The processing time according the Lipschitz value

According to this value, the processing time and the precision of an image are changing. There is a trade off problem. If the Lipschitz value is big, the processing time is very long and the precision of an image is very good. If not, the time is short and the precision is bad. Then, we show the sphere's graph of the processing time and the Lipschitz value in Fig. 9. In a sphere model, the processing time is changing according to the Lipschitz value. However, the precision of the image isn't changing. Therefore, we consider that simple models as a sphere don't need to change the Lipschitz value. We also show the tiger's graph of the processing time and the Lipschitz value in Fig. 10. In a tiger model, the processing time is changing
according to the Lipschitz value and the precision of the image is changing a little. Therefore, we consider that complex models need to change the Lipschitz value in order to increase the precision of the image, although the processing time is too long.

**Figure 9.** The graph of the processing time of a sphere model. When the Lipschitz value is 1, the processing time is set to 0, because the model isn't render.

**Figure 10.** The graph of the processing time of a tiger model.
Chapter 6

Conclusion and Future Work

In this work, we studied artistic rendering of 3D function-based models in HyperFun using ray-tracing as the basic algorithm. In order to generate non-photorealistic image for pen or pencil sketch style, we divided the model projection on a 2D plane into three parts:

1. silhouettes
2. interior
3. background.

If a ray-surface intersection point is found, we apply a rendering technique for silhouettes (chapter 2) or for interior shading (chapter 3). If the ray does not hit the surface, the pixel is rendered as background.

Clear silhouette can be extracted in our experiments from simple models such as a sphere and a cube. However, with complex models, there were parts, where the algorithm cannot extract silhouettes clearly with the width of the silhouette area changing uncontrollable. We consider that only normal vector direction analysis is not enough for extracting silhouettes. We have to improve the extraction of silhouettes by adding the curvature analysis, since this problem often occurred in flat and slightly curved regions of shapes.

Let us discuss three techniques for interior shading brought to consideration in chapter 3. With standard $8 \times 8$ dithering, satisfying results cannot be obtained. The reason is that the use of the fixed dither matrix causes regular pattern appearance in the image. In other words, the intensity of the adjacent mega-pixels isn’t changed, so there is the sequence of same patterns. With random dithering, since a variable dithering matrix was used, the problem with the regular patterns was solved. The reason is that each element of the dither matrix is changed, even if the adjacent intensity isn’t changed. Therefore, there is almost no the sequence of same pattern. Although strokes are not generated with this technique, we can obtain images close to artistic rendering using pencil dots shading. Moreover, we
tested various values of the variation range of each element of the dither matrix. The value of 10 the variation results in the most convincing images with random dithering. The reason is that the image hardly changes if compared with the standard dithering, if the value of the variation range for the dither matrix is too small, and it will become a grayish image if the value is too large. However, satisfying results are not obtained with random dithering from the viewpoint of the pencil sketch style. With generating strokes for interior shading, we could render images similar to pencil sketches. The reason is that not only strokes can be generated, but also the correct direction of strokes can be reliably estimated. However, we could not generate acceptable strokes shading for models such as a cube only having flat parts with almost constant direction of the normal vector. This problem is the same as the silhouette’s problem, because strokes are generated only according to the normal vector direction. We have to improve the generation technique of strokes by adding other algorithm, since this problem often occurred in flat part.

We discussed the processing time of simple models and complex models in chapter 5. In simple models, we considered that simple models don't mean changing the Lipschitz value. The reason is that even if we raises the Lipschitz value, the precision of the image isn't changing and the processing time is only too long. In complex models, the Lipschitz value gives the procession of the image and the processing time great influence. It is because that when the value is raised, the processing time is too long, but the precision of the image is better. The thing that we have to improve is to shorten the processing time too.

This work concerns only rendering in the pencil sketch style in two colors – white and black. We consider it possible to render images in the color pencil style and in the watercolor painting style using ray tracing. Our future work is to render various stroke types such as brush strokes, and, in general, rendering of artistic images using various materials and artist tools. Thus, we have many works to do.
References


