A USER INTERFACE FOR DESIGNING ARTISTIC RENDERING EFFECTS BY USING A LIT SPHERE

LitSphereを用いてアーティスティックなレンダリング効果をデザインするユーザーインターフェイス

by

Makoto Nakajima
中嶋　誠

A Senior Thesis
卒業論文

Submitted to
the Department of Information Science
the Faculty of Science, the University of Tokyo
on February 16, 2010
in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science

Thesis Supervisor: Takeo Igarashi　五十嵐　健夫
Associate Professor of Information Science

ABSTRACT

This paper presents a user interface which makes it easy to design artistic rendering effects of 3D objects by using lit sphere. The original lit sphere system uses reference images prepared beforehand for rendering. Therefore, it tends to be hard to achieve desired rendering effects, and users need to move back and forth between 2D image editing and 3D rendering to obtain the desired result. We propose a system in which the user interactively paints the lit sphere and the system simultaneously updates the rendering result. This makes it more intuitive and easier to design individual artistic rendering effects. In this paper, we explain the user interface and implementation of our system together with the result of informal user tests.

論文要旨

本論文では LitSphere を用いることにより、ユーザが 3D オブジェクトに対してアーティスティックなレンダリング効果を簡単に指定することができるインターフェイスを提案する。オリジナルの LitSphere では、リファレンスとしてロードするスフィア画像のデザインはオフラインで行われていたため、その効果を 3D オブジェクトに適用した際に望みの結果を得ることが困難であり、アーティストが意図した効果を作成するため微調整を繰り返す必要がある場合などに多大な手間が必要であった。本手法では、ユーザが LitSphere に対してブラシストロークにより描画操作をした結果がリアルタイムに 3D オブジェクトのレンダリングに反映されるため、個性的なレンダリング効果をきわめて直感的にデザインする事ができる。本論文では、提案するユーザインタフェースとそのアルゴリズム、およびユーザテストの結果について説明する。

Acknowledgements

I would like to thank Associate Professor Takeo Igarashi for his great and helpful advices. I also appreciate the members of Igarashi Laboratory for their active discussions.

Contents

List of Figures

Chapter 1

Introduction

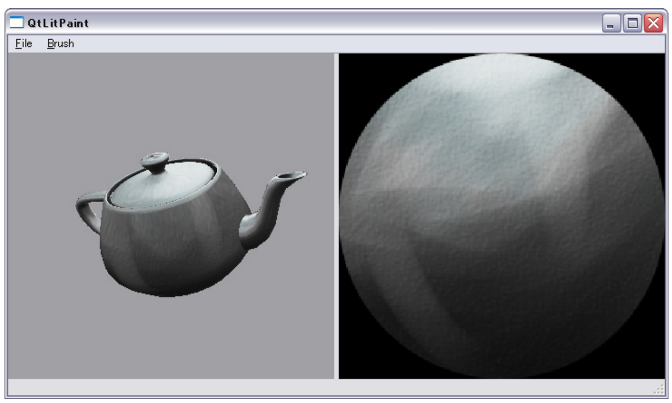

Figure 1.1: An Image of Our LitPaint Interface

1.1 Motivation

To think about dividing the computer graphics industry into two parts, it seems natural that we can separate it between 2D and 3D according to their means of fabrication. Because of great improvements of softwares, the digital 2D computer graphics creating prosess came closer to the analog way, therefore users now can control details of art works intuitively and can create great quality of works easily by computer rather than the past days.

On the other hand, 3D computer graphics creation still needs so many complex operations, and users need to be familiar with the massive softwares. It can be assumed that 3D computer graphics has obvious advangtages, for example, in creating movies, saving labors, or rendering too magnificent scenes. Nonetheless it is hard to say 3D computer graphics technologies are used adequately by people on the street, because it needs a lot of particular practice.

For an instance of this issue of 3D computer graphics industry, we especially pay attention to non-photorealistic rendering technologies. In these days, we can find that several useful non-photorealistic rendering techniques are exist, however, there still are some problems which you may feel when you try to create expressive and artistic rendering effects by them.

The first problem is that setting up visual qualities of effects not by an instinctive interface but by numerical parameters is a too much bother and make it hard to achieve desired results. Actually for this paper we investigated about a famous toon rendering shader softwares, they can create very beautiful and artistic rendering effects but we feel too complex from the way it required for us to adjust its parameters only by numerical select boxes. People often use 3D technologies in order to save labors, and this aim can be reached if the desired result is relatively simple. However, when the desired results are too complex and higher quality is needed, they frequently go off on their primary objective to save workers because it takes too much labors to go back and forth between setting parameters.

The second problem is also related to user interface. In some new revolutionary softwares, you may find that they equip completely original user interfaces in order to make it useful for users. But they tend to be too unique and users are required to be familiar with its specific usage by spending time. To say about these types of user interfaces, users can't apply the way how they use the software to the other one. In these days, there seem to be great progresses in the 2D digital painting works. It can be assumed the reason of this phenomenon is that the analogue-like user interfaces of later softwares made users be able to apply their know-hows which they practiced and learned from real pen and papers to the digital creating process. Starting from well known things is more efficient than starting what we don't know at all, therefore it is important that how easy it is to apply lessons and techniques which our forefathers have accumulated. This point makes much effects on the quality of what we finally create.

Taking these two problems into our account, the essencial factor of creative user interface is that users can control the elements by intiutive and similar way how they do in other situation.

1.2 Our Research

To the purpose of achieving this essence, in this paper we create two softwares, then we show the result images and feedback feelings from them. We create both of our products in order to serve intuitive, friendly and analogue-like control methods for users.

The first function which our sowtware equips is to create artistic rendering effects of 3D models by using lit sphere. The original implementation of lit sphere software aimed to grasp the essence of old arts which is drew by old famous artists who already died, and to apply its rules of the essence to 3D rendering effects. To some extent, this function is useful enough to imitate existing paintings, but when we want to create completely new effects which we have never seen, this interface is less efficient. Other famous 3D composite softwares also have the function like lit sphere, but because in such 3D softwares the process of creating reference images is separated from 3D rendering process, we need to back and forth different tasks. To improve this problem, we attach a standard painting interface to a 3D lit sphere rendering display. This makes it easy to quickly confirm how their operations do effects on finally rendered images, as a consequence, users can get closer results to what they are desired.

The function of our next product is to create hatching lines and shade the image by them automatically accoding to the user's input. This program aims to search the capability of a method which covers the weak point of previous one. By using the algorithm of lit sphere, when a user paints by blurred brush, a color information is well reserved and the rendering result image has good quality. However, when a user paints by solid and sharp brush such as ink pen, because lit sphere doesn't reconstruct the figure of the user's stroke, the result image has undesired rendered surfaces where looks fragmented. To reflect the user stroked hatching line, our program works uses only color brightness in 2D space instead of using 3D structure information. This enables us that user simply draw the line and rendered image truly reflect what user stroked.

In this paper, we present the methods which is used in our softwares, result images, and resultant perceptions.

Chapter 2

Related Work

In these days, non photorealisic rendering has became a stable genre of computer graphics. Naturally, there are a lot of preceding researches in the academic fields which is near at our interest.

2.1 Rendering Techniques

One of the main subject of this paper is to create artistic rendering effects of 3D object. We can find following researches which are related to this point, to say other words the technologies of non photorealistic shading.

2.1.1 Cartoon Shading

A representative of non photo realistic rendering techniques is cartoon shading. Lake, Marshall, Harris and Blackstein researched basic ideas of cartoon rendering [8]. In recent products of computer graphics, we can easily find its applications. This technology aims to render surfaces by sharply-etched colors. For example, by photo realistic rendering a lighted sphere is shaded and colored by gradation, but by cartoon shading it is colored by simply two colors - dark or blight, and you can find the obvious boundary between the darker region and blighter region. This creates cel-animation like results and stylized regions of which are have similar brightness and colors. There are two merits of using this method. The first is that cartoon shaded image can be easily combined with cel animations which human hands made by old style. Cartoon shading have not yet been efficient to create all the process of cel animations only by it, therefore professional animators create important parts of the animation and in relatively easy parts automatic systems of cartoon shading switches their works. In this paragraph, the term ”easy” means that the shape of target objects straightly reflect its real existing shape, or that the scene has simply repeating or gradually continuous movings. The second is that simplifying makes it easy for viewers to concentrate on what is really important points. Because photo realistic rendering tends to render equally the details of all things, frequently users are buried in a lot of informations and lose the sight of what is important. So simplifying by cartoon shading is useful in the situation such as visualizing or explaining complex phenomena.

2.1.2 Hatching

Hatching is a term of the art drawing. A painter fill the shaded region by slight lines which are have almost same directions. This is a basic technique of drawing, however this is pretty useful to present textures of the target object and to express warmful actions which the painter did. Therefore, there is a lot of preceding researches about hatching [1] [6] [9]. The researches can be divided into mainly two patters by its input data (two dimensional input or three dimensional input). By two dimensional inputs, they process the two dimensional color information, compute the difference of brightness and sometime using edge detective algorithms to determine where they paint by hatching lines. The direction of lines are almost controlled by user inputs. On the other hand, the 3D oriented ones are uses the information of target's shape, the angle between the viewpoint and the target, and normal vectors of the surface. This has more efficient expression of the gradient of the target's surface. Though the way how the lines curve reflects the geometry of the target model, a viewer can easily observe how it shapes. But this method often resulted in such as contour lines, it seems more mechanical style than hand-drawn style.

2.1.3 Tone-based Shading

Usually shading means creating gradations from object color to black based on how it is lighted. On the other hand, Gooch et al. propesed tone-based shading which uses not black colors for shading [4]. Its luminance and hue are used to decide how to blend the object color and specified color. This can be used, for example, in expressing the concept of a temperature of the surface. The gradation from blue to red seems to be the gradually changing its temperature from cool to warm. Of course this doesn't truly reflect its physical phenomenon, but this is useful to create impressive rendering effects and make it easy for users to grasp the shape and geometry of the object.

2.2 Researches to Extract Individuality

And as the other aspect of our interest, we can find some approaches which is designed to grab the users' individuality.

2.2.1 Graftals

Kowalski developed art-based interface for rendering detailed object [7]. This is the technology to allocate the strokes which user inputted, on the surface of 3D models. By using this technology, it is made relatively easy to rendering objects which has may furs, such as soft toy of teddy bear, glass, or tree. Because, there's no need of modeling all complex scenes. Graftal system automatically allocate fine parts on the simple surface of 3D objects, and finally complex image is rendered. The interesting point of this technology is that even though the last result is composed by computational automatic process, but each elements are crated by a user and they directly reflect what the user draws, as a consequence, the result image can reflect what the user intended. This is a good example to learn instinctive user interface is important to get results be able to reflect user's mind.

2.2.2 Direct WYSIWYG Painting

A direct interface for painting edge lines and textures was developed by Hanrahan and Haeberli [5]. This mainly has two functions. The first is that user can simply draw illustrations on the surface of three dimensional model. In general, texture of 3D model is created by extracting its surface such as UV mapping. So it was hard for beginners to achieve desired textures by using UV map, but this product supply an intuitive interface to enable that a user simply draw illustrations on the model with confirming what the result is at the same time. The second function is that user can control how the boundaries are rendered. The user draws a brush stroke, then the system automatically apply its color, style, and shape to the way how to render target 3D model. If the user draws black, solid and straight ink-pen stroke, then the boundaries of target object is rendered in being rounded by black and pen-like lines. If the user draws dotted one, the result seems so. WYSIWYG has already became a worm-out term, but it is still important especially in the products which relate to users' creativity.

2.2.3 Lit Sphere

Sloan, Martin and Gooch developed Lit Sphere technology [10]. The purpose of this technology is to emulate expressive effects found in works of great artists by sampling their works. In a general way, colors of the rendered objects are decided by how they reflect the lights. The lit sphere creates a reference sphere which represents how the target object reflect the light and how to be shaded, then apply the reference sphere to render complex objects in the same way how the reference is. The merit of this approach is that a user can concentrate on color and shading only, and the user once created the reference sphere then he can apply it to many objects in various situations. In addition, there are an useful interface is introduced. We can use the interface to sample from existing art works and seamlessly compose it to a reference sphere. In their paper, they sampled a Cezanne's art work for reference sphere and by using it they rendered 3D models which seems like to be created by Cezanne. This approach is useful enough to capture essences from existing art works.

2.2.4 Analyzing Individuality of the Artist

Cole researched about how artists pick up and put off the apparent lines of the target object when they draw art works [2]. They focus on the curvature factor of edges, then analyze that what type of edges are drawn by artists. According to the gathered data of artists' characteristics, a imitation model is developed. This model draws the line art work which imitates the original work of the artist. This attempt seems successful and there are next research releted to this [3]. What the important point of this work is not only its results to be able to imitate art works of artist, but also its method of the research. Because the evaluation of the artistic product differs according to emotion of every person, it tends to be hard to construct common framework to evaluate art works. In a similar way, it is also hard to extract the essence of every art works and properly compare them, because the art works often are too unique to compare mutually. Therefore, they determine a minimum framework to restrict art works without losing artistic individualities completely. Then, in this mechanism they gather data and statistically processed them. We can learn a lot from the way how they test the artists without too much uniforming them.

Chapter 3

Overview

In this paper, we present two user interfaces which aim to assist that user can create artistic effects intuitively.

3.1 User Interface of LitPaint

The first product consists of lit 3D rendering component and brush painting component. In this paper, we call this software as LitPaint.

On the painting area, a user can assign colors of the reference sphere by brush, then our software simultaneously apply the input data to the 3D model and render it by sphere mapping algorithm.

This enables users to see how the reference sphere which they painted is applied at the same time, and make it easy to edit reference sphere in order to achieve the effects they imagine.

3.2 User Interface of AutoHatching

The second product can automatically create hatching lines which is oriented to user inputted, only from a two dimensional background image. The role of this program is to cover the weak points of previous lit sphere product . We call this software as AutoHatching.

The usage of this program is that a user drag a line where he wants to draw hatching line, then our program cover the appropriate regions by the lines which preserve the direction and the style of what user drew by hand. When the background image is changed, hatching layers are reconstructed and rendered again by orienting each inputs.

By using this interface, users can easily create a lot of hatching line from a little input.

Chapter 4

LitPaint

In this section we talk about the implimentation of our LitPaint software. To say shortly, what this software do is that to decide what color is used to the purpose of shading the surface of a 3D object according to its normal vector of every vertex.

4.1 Determine Colors by Normal Vector

At the beginning, we consider what makes colors of what we see. In modern theory of color, a color is divided into three elements (hue, lightness, and saturation). When we see an object, the hue of its color is decided by the material of its surface and the color of the environmental light. Thinking about the situation we see a mature apple on the sun light, the surface of the apple well reflects red rays than green and blue, therefore we see the apple is red.

On the other hand, are remaining two elements of color - we combine them and call it as tone - decided by what? We can observe tones as color gradations on the surface of objects. Children's coloring works often ignore this changing of tone, so they use a simply named color to paint an object on the canvas, for example, sun is red, sea is blue, and tree is green. But we adults know that even if a object consists entirely of one material, it doesn't seem only one color.

The tone of the color differs according to the intensity of the light. A light which our eyes receive is what the surface of the object reflects. Then the intensity of the reflected light, according to the theory of ray reflectance, depends on the cosine component of the angle of incidence. Therefore we can say that the tone of the object depends the angle of its surface.

The angle now we talk about is the degree relating the light source, the target coloring point and the view point. As a corresponding value with this angle, we think about the normal vector of the target point.

4.2 Surface Normal and Vertex Normal

There are two types of normal vector, surface normal and vertex normal. Surface normal represents the gradient of the surfece, is calculated from outer product of its vertices. Because a polygon has one surface normal, when user specify using surface normal, the polygon is shaded by one solid color.

On the other hand vertex normal can be used when the user want smooth shading of the polygon. It can be calculated in the way that a vertex gather polygons which includes the vertex, then the surface normals are summed, at last by normalizing the sum we can get the vertex normal. By using vertex normal, normal vectors in the inner area of the polygon is interpolated by calculation, then we can see gradually shaded polygon. After this, when we talk about normal vector, it means the latter one, vertex normal.

Because a normal vector represents how leaned the surface is, we can consider that if two points have same value of normal vector, they are observed as the same tone of the color. The main point is that we can treat the color tone of a point according to its normal vector. On the basis of this idea, lit sphere decides how to render the 3D object. At first the reference sphere is prepared, then, according to every normal vector of rendered polygon's vertex it searches matching points of them on the reference sphere. A patch is determined by the matching points and clipped, at last, the patch is used as texture of the polygon.

4.3 Environmental Mapping

We gave an explanation of the theory of lit sphere, in the next place we talk about how to implement it. On the original paper, about the way how to implement it they only say ”environmental mapping hardware is hired” [10]. Environmental mapping is a common word in 3D computer graphics, but we need more explanation why this can be used for lit sphere rendering, to say other words, why this is enough to find matching points according to normal vector.

Environmental mapping includes some pattern of its method, in this case we adopt sphere mapping. Basically, sphere mapping is a technique to express mirror reflectance by using the value of normal vector as the coordinate of the texture directly. The calculation of texture coordinate value is that x and y component of the normal vector, which is normalized and is in the range of [-1, 1], is scaled into the range of [0, 1] and used as the texture coordinate. This calculation is done in every vertex and every polygon use it and clip the texture to render it. This needs only simple scaling calculation in every vertex so it's not much complicated.

We told that the angle between the normal vector and the viewpoint vector is important to decide the tone of the object. Therefore if we set the viewpoint on the front of the object, sphere mapping exactly means where is the matching point on the reference sphere with the target vertex of rendered object. Our program implements this method by using functions of OpenGL.

Chapter 5

AutoHatching

In this section we talk about the implimentation of our AutoHatching software.

5.1 Object Space and Image Space

What our primary purpose of this software is that we hope the inputted hatching lines to be displayed on rendered image without being transformed.

By lit sphere approach, input strokes on the reference sphere is once chopped and transformed according to the shape of the polygon, therefore we see distorted hatching lines on the resultant image. Such attempt to avoid coherence breaking of hatching lines is also surveyed by Coconu [1]. But he discussed about this problem on the object space, which means 3D space.

To think about the situation when we draw hatching arts by pen and paper, does the 3D information of the object is really needed? For my instance, I project the image of target object into 2D screen on my brain and then draw hatching lines on the region which has the same tone. So we think we might be able to draw hatching lines only using image space information, which means 2D information.

5.2 Region Selection

The main factor of this software is how to select the region which has the same tone. Especially the interface is important to enable that users can intuitively control the effect. This time we hire following method.

At first the user drag a hatching line on the above of the image. Our interface tracks the motion of user drawing, and find maximum and minimum value out from the points where the stroke passed. This maximum and minimum value of color tone are used as a range of selected region, and we create alpha channel mask according to this range of color tone and background image. Then we create a image which is paved all the screen with user inputted hatching line. At last we clip the paved image by alpha channel mask, then render it neatly.

The merit of this interface is that a user don't need to worry about numerical values of tones. What they need is to simply draw strokes wherever they want. Then our interface automatically calculate where the user want to fill by one pattern of hatching line and render it. This require nothing special to users, simply save the routine works of them.

Now users can control desired effects without fighting with some dry numerical parameters.

Chapter 6

Results

6.1 Resultant Images of LitPaint

We propose the resultant images of LitPaint, see figure 6.1 and 6.2. Figure 6.1 is a result of using our LitPaint software. On the contrast, figure 6.2 is a result of the attempt that reference sphere is created by off line, then applied and render the resultant image. To be honest with these results, we can see only a few differences between them. We consider the reason of this is that the expressing ability of the brush we implemented in LitPaing interface is too limited for users to exert their ability of painting. The low function of the brush limits the capability of painting, therefore we need to improve this point.

As to say about other point of attention, when we use the reference image which painted by large width and blurred brush, the rendered image seems to have good quality (see figure 6.3). On the other hand, when we use solid and thin brush, there seem to be messed rendering image of the surface (see figure 6.4). After all, lit sphere algorithm is good for deciding what color to render, but not good for reserving the shape of the user stroke.

And some reference image resulted in viewers feel so much that the object is made with mirror reflecting material (see fugure 6.5). This phenomenon is sensible when there are sharp edges of color in the reference sphere. If there are edges in the reference sphere, when we change the view point, we see the rendered edge moving on the surface of the object. So we feel the object is mirror reflecting than it is shaded.

6.2 Resultant Images of AutoHatching

Then we propose the resultant images of AutoHatching, see figure 6.6 and 6.7. It seems good for every result. The purpose that to preserve the inputted stroke seems to be accomplished. However, there are some cases such that the rendered image is good when we see the 3D object from a stable view point, but the resultant image runs off from whate we desired when we change the viewpoint.

This problem can also be found on LitPaint program. The purpose of our developing these technologies is to crate artisitic rendering effects being used on movie. Even so the result is good in a single image, it is no worth if the result is far from what creators desire when we use it in movie creation. This indicates us not only that we need more improvement, but also we do need a interface which enable users to confirm resultant image simultaneously in order to avoid making undesired results.

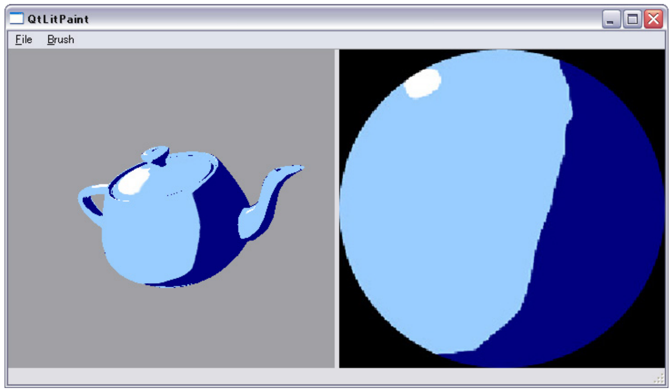

Figure 6.1: Using On-line Created Reference Sphere

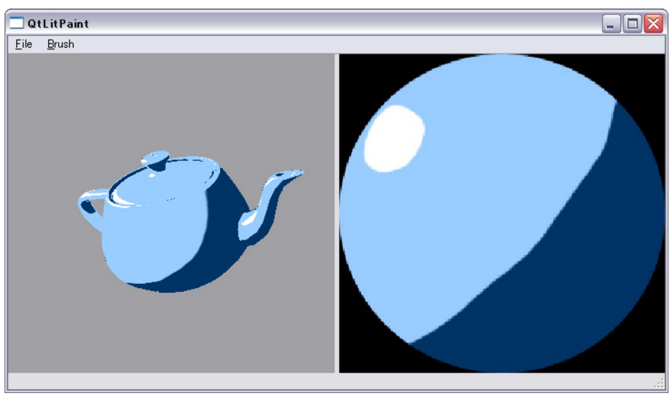

Figure 6.2: Using Off-line Created Reference Sphere

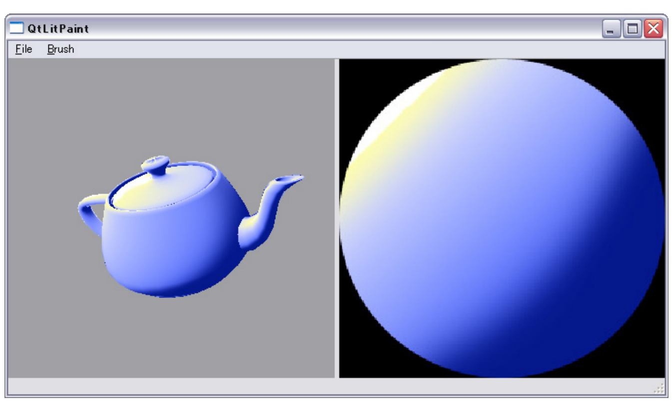

Figure 6.3: Drawn by Brurred Brush

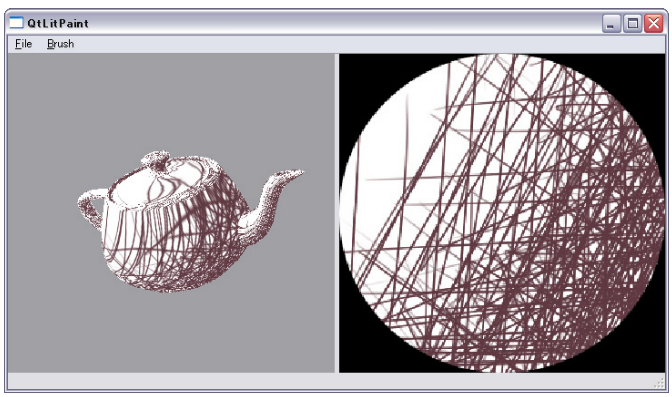

Figure 6.4: Solid Edges in Reference Sphere

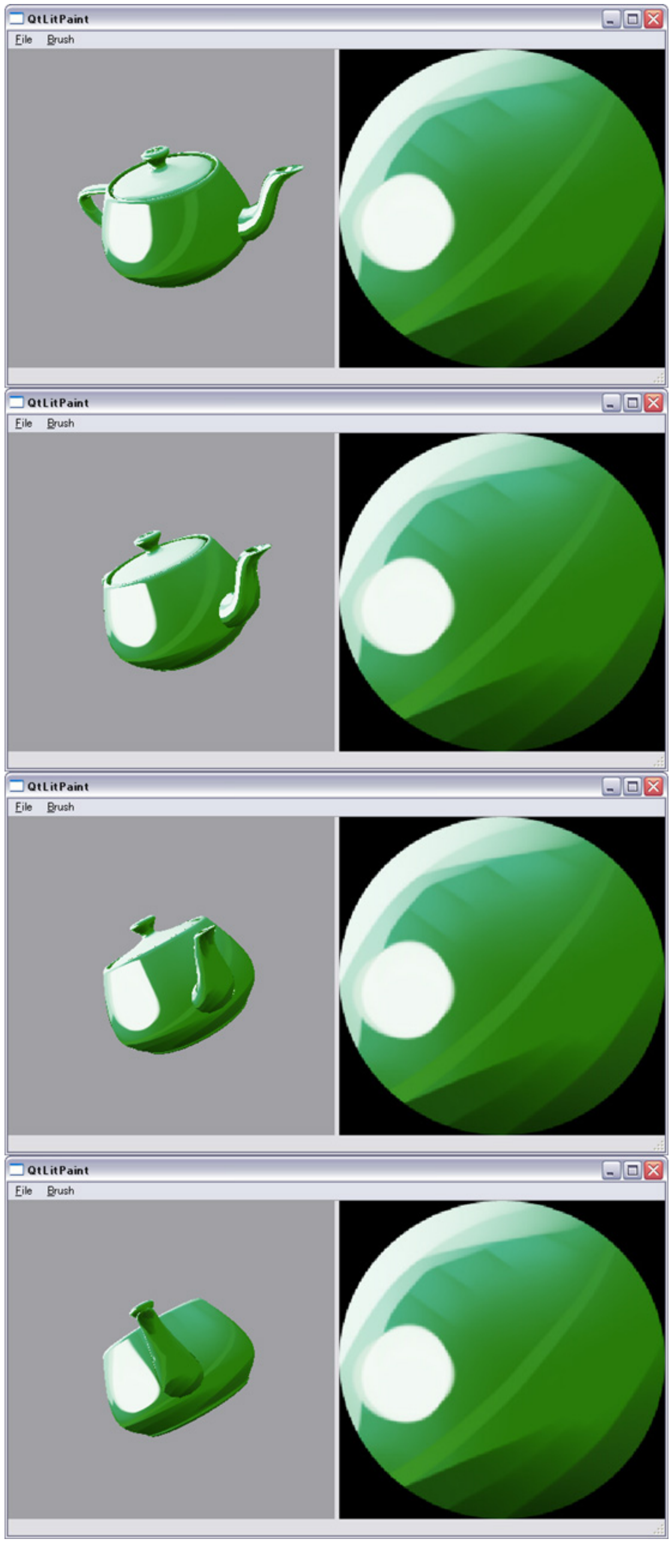

Figure 6.5: Rotate the 3D Model

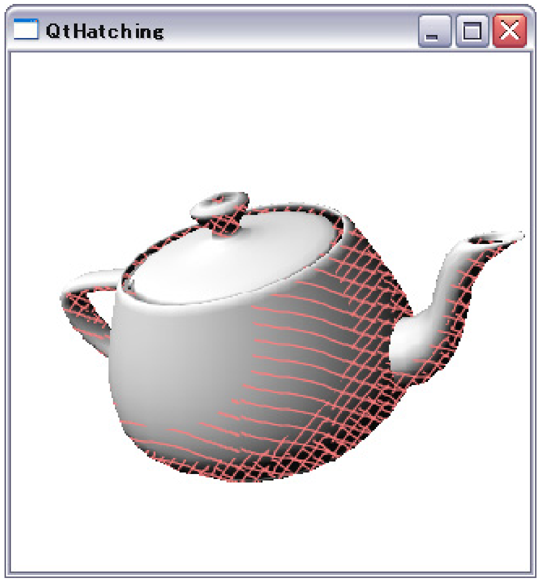

Figure 6.6: Result of AutoHatching

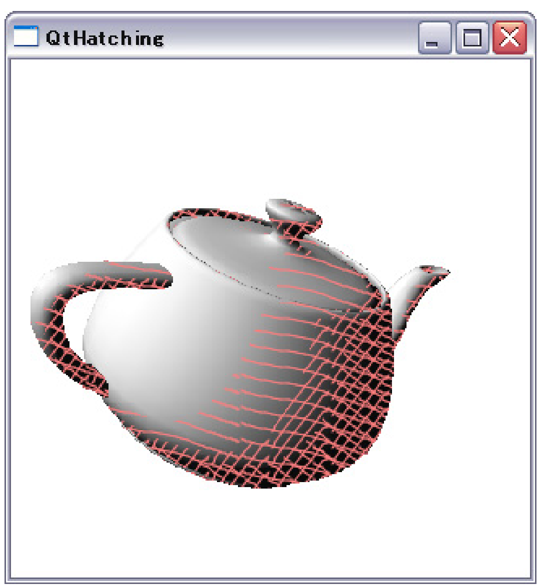

Figure 6.7: Apply the Same Rule to Another Image

Chapter 7

Conclusion and Future Work

Viewing from advantageous standpoint, the interfaces which we created are useful enough to make it easy to create complex and artistic rendering effects. It is a meaningful advantage that users don't need to be aware of numerical parameters, but only to draw strokes in the same way how they have ever done by analogue pen and paper. By these interfaces users can concentrate the matter of what they create without bothering any mechanical points, and thus the quality of the product which they finally create is improved.

However, it seems true that we have some limitations which is closely concerned with the quality of products. You may have a look of resultant pictures in the previous section, and it can be said that there are actually good result from a viewpoint where user worked, however from another view point there are some feelings of strangeness even though the same rule are applied to render the object. That is to say it makes some strangeness on different viewpoints to simply apply the same rule which only accounts the direction of normal vector and the brightness of every pixel. Humans probably change their internal drawing rules unconsciously, but our software can't do it in the same way artists do, so this leads to make some unreasonable resultant images. To be honest with our research, it is not enough to decide how to render 3D objects only by the direction of normal vector and color brightness of its surface. This is what we learned from our research.

Then we have demands to concern about what information is required to create really desired rendering effects. Although these are hypothesizes yet, we could count three important elements.

Relative Displacement with Other Elements When we draw illustrations, we take it into account that the percentage of the color and the position in the canvas. It is important that the amount of the value indicating how the element differs from others.

Coherence with Neighboring Elements Sometimes we want to emphasize a difference between the elements on the canvas, but other time the continuity is important. To say in other words, when we try to decide what color is used for a pixel, we need to concern about not only the pixel itself but also its neighbors.

Stylize and Simplify the Primitives Now we treat single pixel as a unit, but in our brain we normally figure out the target's shape in the stylized and simplified form. Thinking the situation that we draw a dice, we don't put every pixel point by point, but we put some shapes we see such as square, rhombus and circle. In the computer graphics, this type of abstraction seems to be needed.

Based on the importance of these ideas, we want to continue our researches about such technologies.

- Interface which capture and save the individual information about above points without any special operations
- Easily cooperating rendering method to apply the information which is gathered by previous one

We hope that, in the future, we will be able to preserve the individuality of art drawing and everyone can apply chosen one to render 3D movies.

In addition, we propose some future work about implementations of the software which we created this time.

Our LitPaint software has a limitation that if the position of light is changed, our model can't apply the chenging of light. Because now we preserve the input date in 2D reference sphere, therefore we can't take the information of light source position into account by truly meaning. In order to address this problem, it can be considered that the reference sphere is preserved as 3D with the information of light position, then render it to 2D image and use it to sphere mapping. When the light position is chanded, the 3D reference sphere is roteted and rendered for new reference image. This will enable the lit sphere method to follow the changing of the light source position.

Another improvement we think is that we make it possible to intuisively adjust the margins of hatching lines on our AutoHatching software. Artists often express the tone of the object by the density of hatching lines. However, because current implementation of our AutoHatching software uses fixed amout of the margin, so if a user want to more dense pattern of hatching, he shoud superimpose a new line as the same tilt with underlying lines. This improvement will make it more easy to express the tone differences by haching.

References

[1] Liviu Coconu, Oliver Deussen, and Hans-Christian Hege. Real-time pen-and-ink illustration of landscapes. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 27–35, New York, NY, USA, 2006. ACM.

[2] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[3] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–9, New York, NY, USA, 2009. ACM.

[4] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452, New York, NY, USA, 1998. ACM.

[5] Pat Hanrahan and Paul Haeberli. Direct wysiwyg painting and texturing on 3d shapes. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 215–223, New York, NY, USA, 1990. ACM.

[6] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[7] Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden, and John F. Hughes. Art-based rendering of fur, grass, and trees. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 433–438, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[8] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 13–20, New York, NY, USA, 2000. ACM.

[9] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, 1990.

[10] Peter-Pike J. Sloan, William Martin, Amy Gooch, and Bruce Gooch. The lit sphere: a model for capturing npr shading from art. In *GRIN'01: No description on Graphics interface 2001*, pages 143–150, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society.