การให้แสงและเงาแบบจำลองสามมิติในกระบวนแบบลายเส้นพู่กันจีน

นางสาวนุชรี ทองทั้งวงศ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555

3D-MODEL RENDERING IN CHINESE BRUSH STYLE

Miss Nucharee Thongthungwong

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science and Information Technology
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2012

Thesis Title	3D-MODEL RENDERING IN CHINESE BRUSH STYLE
By	Miss Nucharee Thongthungwong
Field of Study	Computer Science and Information Technology
Thesis Advisor	Rajalida Lipikorn,Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

…….. Dean of the Faculty of Science
(Professor Supot Hannongbua,Dr.rer.nat.)

THESIS COMMITTEE

……. Chairman
(Assistant Professor Suphakant Phimoltares,Ph.D.)

…….. Thesis Advisor
(Assistant Professor Rajalida Lipikorn,Ph.D.)

……. Examiner
(Assistant Professor Nagul Cooharojananone,Ph.D.)

……. External Examiner
(Associate Professor Viwat Udompitisup)

นุชรี ทองทั้งวงศ์ : การให้แสงและเงาแบบจำลองสามมิติในกระบวนแบบลายเส้นพู่กันจีน (3D-MODEL RENDERING IN CHINESE BRUSH STYLE) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. รัชลิดา ลิปิกรณ์, 25 หน้า.

การให้แสงและเงาแบบจำลองสามมิติ กำลังเป็นที่นิยมในวงการภาพยนตร์รวมถึงภาพยนตร์ประเภทการ์ตูน โดยที่ผ่านมามักจะเป็นการให้แสงและเงาแบบจำลองสามมิติในรูปแบบที่ดูสมจริงจนแยกออกได้ยากว่า ภาพที่เห็นนั้นเป็นภาพถ่ายจริงหรือเป็นภาพที่สร้างขึ้นด้วยคอมพิวเตอร์ แต่กระนั้นก็ตามก็มีอีกแนวทางหนึ่งในการให้แสงและเงาของแบบจำลองสามมิติโดยมีความต้องการที่จะให้ภาพออกมาดูเสมือนภาพวาดในเชิงศิลปะรูปแบบต่างๆ ซึ่งการที่จะให้แสงและเงาแบบจำลองสามมิติให้ออกมาดูเหมือนภาพวาดนั้น จำเป็นต้องเริ่มจากการหาเส้นที่จะเกิดบนแบบจำลองสามมิติที่จะแสดงวัตถุนั้นในรูปแบบสองมิติได้ แต่กระบวนการคำนวนหาเส้นดังกล่าวนั้นใช้ทรัพยากรและเวลาอย่างมาก เพราะมีการคำนวนที่ซ้ำกันเกิดขึ้นมากมาย อีกทั้งผลที่ได้จากการคำนวนนั้นเป็นภาพรูปร่างของเส้น ซึ่งยากที่จะนำไปพัฒนาให้มีกระบวนแบบต่างๆ ในเชิงศิลปะ วิทยานิพนธ์ฉบับนี้นำเสนอวิธีลดการคำนวนที่ไม่เกิดประโยชน์ และการสร้างเส้นที่แสดงแบบจำลองสามมิติที่สามารถนำไปพัฒนาต่อเพื่อสร้างกระบวนแบบในเชิงศิลปะ โดยมุ่งเน้นไปที่กระบวนแบบลายเส้นพู่กันจีนเป็นประเด็นสำคัญ

สาขาวิชา: วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ลายมือชื่อนิสิต .....

ปีการศึกษา 2555 ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

5273606023 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS : NON-PHOTOREALISTIC RENDERING / 3D OBJECT / ANIMATION / LINE RENDERING / SILHOUETTE

NUCHAREE THONGTHUNGWONG : 3D-MODEL RENDERING IN CHINESE BRUSH STYLE. ADVISOR : ASST.PROF.RAJALIDA LIPIKORN Ph.D., 25 pp.

3D rendering has become popular in the film industry. However, for the animation industry, they tend to find a method for rendering in artistic style. There are so many approaches for creating stylized rendering. One of which is to create traditional animation's look, which has edge lines or silhouette edges. In order to succeed this idea, it's necessary to find silhouette edges on 3D model. However, calculation for finding silhouette edges on 3D model is time and resource consuming because of redundant calculation. Also the results of calculation mostly are images or fragments of line, which are hard to stylize. This thesis presents a technique for reducing redundant calculation and stylize silhouette line focusing on Chinese brush paint style.

Department : Mathematic and Computer Science Student's Signature .....
Field of Study : Computer Science and Information Technology Advisor's Signature .....
Academic Year : 2012

ACKNOWLEDGEMENTS

I would like to express my great appreciation to my advisor, Asst. Prof. Rajalida Lipikorn who generously offers guidance and support during my study here. She helps me a lot in understanding how mathematics formulation works on computer, also how to do object oriented programming. I would also like to thank to Asst. Prof. Nagul Cooharojananone, who taught me on algorithm and data structure which play important roles in this thesis. My grateful thanks are also extended to Asst. Prof. Supakant Phimoltares, his Computer Vision class helps me a lot in comparing computer's perception of image with artist's perception, and to Assoc. Prof. Viwat Udom pitisup who expands my vision of Chinese brush painting style. Without their support, the completion of this thesis would not have been possible

CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

3D rendering has become popular in the film industry. However, for the animation industry, they tend to find a method for rendering in artistic style. This trend is called non-photorealistic rendering. There are so many approaches for creating stylized rendering. One direction is to create traditional animation's look, which has edge lines or silhouette edges. In order to succeed this idea, it's necessary to find silhouette edges on 3D model and do projection on 2D plane as vector lines. Then it can be stylized such as Chinese brush painting style.

Objectives

This research aims at developing an algorithm for creating the Chinese brush style on 3D animation rendering. The objectives are:

1. To generate silhouette lines from 3D model.
2. To create continuity of 3D-based silhouette lines in frame sequence.
3. To render lines in Chinese brush style.

Problem Formulation

1. Extracting lines in 3D model is time consuming, and also the extracted line is hard to control for shading in artistic style.

2. Most recent line shader does not interact with the light source which causes the misunderstanding of light and shape of the 3D model when doing the non-photorealistic rendering.

Scopes of work

1. The 3D models used in this thesis are created by modeling in 3D applications.

2. The 3D models used in this thesis must be non-manifold polygon.

3. The 3D models should have the number of polygons no more than 5,000 triangle faces (or 2,500 quad face), which are considered as medium resolution character model in game industry.

4. Only lines extracted from the 3D models will be rendered (Not render the surfaces).

5. One light source per one shader will be used as a reference line rendering.

Methodology

The methodology for this research consists of 2 parts. The first part is to generate silhouette lines, and another is to create the Chinese brush style rendering.

To generate silhouette lines or edge detection, which can be stylized, the geometry-based approach is considered. Though it might take time for computing, GPU technique could help fixing this problem. Point cloud technique is recommended for storing the lighting data in order to synchronize the edge lines of a changing model in animation.

Moreover, in order to reduce the amount of edge comparison, the geometry data should be collected, especially the edge data. It should be organized as tree data structure, which can create the relationship of each edge so that in comparison level, only related edges would be compared.

The extracted lines from 3D model should be vector lines so that we can apply the line shader, and have the light source for controlling the line weight and thickness.

For the Chinese brush stylization, the 3D brush technique would be adapted to imitate the brush stroke rendering. The light data on geometry vertex might be used for controlling the weight of brush. Also the line direction might affect the 3D brush shape.

CHAPTER II

LITERATURE REVIEW

Basic 3D rendering

3D rendering is the technique to create computer generated 2D image based on 3D modeling. This is similar to taking a photo or filming a scene via camera in real life. Several 3D rendering methods have been developed and also can create various visual presentations such as wireframe, smooth shading, photo realistic and non-photorealistic, etc.

"Rendering" in term of artist is to draw, paint, or present visual art based on artist's perception. Basically, artist's fundamental of perception is to see or to view things. People can see things, because light falls on an object and reflects to human's eyes. Light, in Physics, is a range of electromagnetic radiation that can be detected by human eyes. Therefore, what people actually see is the light distribution on an object, which is called shading in 3D rendering.

In Computer Graphics, how an object is seen on viewing camera, can be explained by ray tracing. Physically, ray tracing is described as: light rays are emitted from light sources, bounce around in a scene, reflecting off objects and picking up their color, and finally strike the film in a camera. However, in 3D rendering, ray tracing does not follow rays outward from the light source, because most of rays do not come to the camera. Also it consumes too much resource to compute every ray from light source, and pick only ones that can make their way to the camera. Instead of following rays from the light source, ray tracing rendering tends to trace a ray from a viewing camera through each pixel on a viewing plane, and calculate color of visible objects. At this state, it is called ray casting. When ray hits a surface, it could produce up to 3 new types of ray: reflection, refraction and shadow. Reflection ray is a mirror-reflection direction from shiny surface, and then it is intersected with objects in the scene. The closest object intersects with the ray will be seen in the reflection. Similarly, refraction ray travels though transparent objects with refractive index. Shadow ray is sent to a source and is used to determine if a point on surface is visible to a light. After ray casting, ray tracing will generate up to 3 new types of ray and calculate the final color of that pixel on screen.

Though ray tracing works well in explaining light effects in real life, it is not enough. As mentioned earlier, physical light is a range of electromagnetic radiation. Electromagnetic radiation can be both waves and particles. Ray tracing explains light distribution as wave pretty well, but it still misses some light effects, for example global illumination, that can be well explained by particle with energy called photon.

Global illumination is the calculation of how light affects an entire scene, especially the contribution of light reflected between surfaces. This is opposed to coming straight from a light source. Jeremy Birn (lighting technical director at Pixar and author of *Digital Lighting and Rendering*, 2006) defines global illumination as "any rendering algorithm that simulates the inter-reflection of light between two surfaces. When rendering with global illumination, you do not need to add bounce lights to simulate indirect light, because the software calculates indirect light for you based on the direct illumination hitting the surfaces in your scene".

Figure 1 Illustration of Physical Cornell Box

As ray tracing generates up to 3 new ray types to determine the final color, global illumination assumes ray, travelling from a light source at a point on surface, would distribute new rays. These new rays would be considered as a new light source for further calculation. The light from distributed ray on a surface is called indirect light. Cornell Box (figure 1) is a good example to explain this kind of light effects.

Created by Cornell University (Goral et al., 1984), Cornell Box is a test aimed at determining the accuracy of rendering software by comparing the rendered scene with an actual photograph of the same scene. A model of the box consists of one light source in the center of a white ceiling, a green light wall, a red left wall, a white back wall and a white floor. White objects are often placed inside the box to show the effects of global illumination. For example, some light should reflect off the red and green walls and bounce on to the white wall so part of the white wall should appear slightly red or green. Also the white objects should be tinted by the color of the wall since the light, travelling from the light source to the colored wall, bounces colored ray and becomes indirect light that would affect the color on object's surface. Sometimes this kind of effects is called color bleeding.

There are several methods to calculate global illumination in 3D rendering. Conventional radiosity is the first method of global illumination to become available. It is assumed that indirect light is transmitted between surfaces by diffuse reflection. The light leaving a surface (its radiosity) consists of self-emitted light and reflected or transmitted indirect light. The amount of light arriving at a surface requires a complete specification of the geometric relationships among all reflecting and transmitting surfaces, as well as the light leaving every surface. Greenberg et al.'s equation (Greenberg et al., 1986) explained that the amount of light energy leaving particular surface is equal to the self-emitted light plus the reflected light. The reflected light is equal to the light leaving every other surface multiplied by both the fraction of that light which reaches the surface in question, and the reflectivity of the receiving surface. The sum of the reflected light from a given surface plus the light emitted directly from the surface is its radiosity. To render an image, the discretized radiosity information is used to create continuous shading across a given surface. This means it is needed to increase more poly count in order to achieve more detail, and if the objects are animated or moving, it needs to be recomputed every frame.

Photon mapping is a two-pass global illumination method developed by Henrik Wann Jansen (Jensen, 1996). The first pass is to construct photon maps by emitting packets of energy (photons) from the light source and storing these as they hit surfaces within the scene. The second pass is to render image using a distribution ray tracing algorithm optimized by using the information in the photon maps. The speed and accuracy of the photon map depends on the number of photons. Photon mapping is used to simulate realistically the interaction of light with different objects, especially the refraction of through a transparent substance such as glass or water.

Non-photorealistic rendering

When talking about 3D model rendering, most people are thinking of realistic rendering as mentioned previously. It's an attempt to create the final image to be like realistic photograph with correct light effects' calculation. There are many researches and developments in realistic rendering that are now using a lot in live action films as well as in animations.

However, the film industry, especially animation industry, is expecting for some stylized rendering. They are expecting for some kind of rendering that goes beyond realistic, some kind of rendering that looks like artist's painting. These are called non-photorealistic rendering. There are many attempts in creating non-photorealistic rendering for 3D objects. For example, in 1996 Disney attempted to create Monet's painting style on 3D rendering (Meier, 1996). This technique creates particle around 3D object, then renders the particle with brush map with stroke direction. Comparing to artistic painting technique, the Disney's technique is similar to volumetric painting technique. The volumetric painting technique is to paint what the artist see in an area of shape. There's no rough line sketch, nor line to define the shape.

According to artistic technique, there is also a technique of drawing line to represent object's shape. This technique is quite popular in traditional animation (hand-drawn animation) and anime (Japanese style animation). In computer-generated animation, there is an attempt to render 3D model to have the final image to be like traditional animation or anime. Cel shading or Toon shading is a very popular approach for creating anime style rendering. It is done by reducing the total number of colors from smooth to a limited range of color palette, and then accentuating the object's borders and lines. The borders and lines in Cel shading can be created by various methods. One popular method is to use back-face culling. Firstly, invert back-face culling (polygon's normal vector), back facing polygon would be rendered as black thick line. Then set back-face culling to normal and render surface with stylized shader. Final image is done by compositing via Z-buffering, as the back-faces are always deeper in the scene than the front-faces, therefore, the object is rendered as black outline with interior contour line. Another method for creating borders and lines in Cel shading is to use edge detection filter. It is done by firstly rendering scene's depth and world-space surface normal. Then use edge detection filter such as Sobel filter to apply to normal/depth image to generate edge image. After that, render color image with

stylized shader. Finally edge image and color image are composited to produce final rendered image.

However, the final rendered image created by Cel shading has some issues, especially the issue about the borders and lines. Are these lines actually representing 3D shape? Do these lines really happen in traditional drawing? This comes to a question of how artist knows which line to draw. Cole et al. (Cole et al., 2008) has done a research of comparing artist's line drawing and 3D object edge detection. The result is that lines drawn by educated artists overlapped almost 75% of 3D object edge detection, particularly detected by Silhouette Edge Detection algorithm. It can be assumed that 3D object edge detection can explain 3D shape similar to artist's line drawing.

There are various approaches for rendering the edge line from 3D modeling which can be classified into three main approaches. The first one is the geometry-based approach. One of the concepts of this approach is to detect silhouette edges by finding the edges, which have both front facing and back facing (figure 2). In order to find the silhouette edges, it is necessary to compare every edge of the geometry. It is time-consuming, complicated and difficult to implement. This approach is also inappropriate to work with changing meshes (animation sequence).

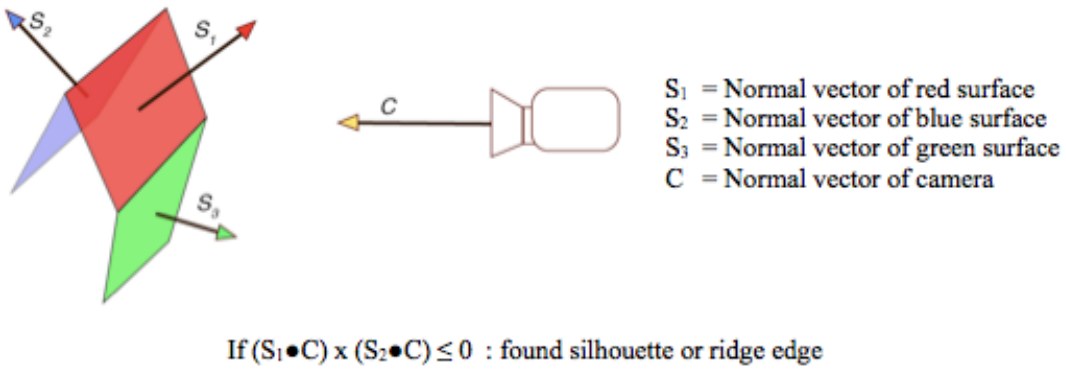

Figure 2 Illustration of Finding silhouette edge

The second one is image-based approach (Gooch et al., 1998), which focuses on depict only the portion of the silhouettes that contribute to the final 2D image. It uses 2D image-processing technique such as edge detection filter, leveling the contrast of light and shadow, and so on. This technique lacks the control for the line thickness, and also hard to generate the stylization of the brush stroke.

The third approach is the hybrid technique (Markosian et al., 2000), which combines the edge detection technique and image-based approach together. It detects edge by geometry-based approach, but stylizes the line with image-based approach. Some of the hybrid approach recommends using randomized algorithm for edge detection in order to reduce time-consumption. This approach still lacks the interaction with the light source. In order to do the line shading or stroke stylization, a hybrid approach is considered since the line extraction from this technique is available to apply the shader.

Brush stylization approach

Physically traditional drawing and painting could be stylized by various approaches such as tool, artist's perspective, artist's skill, etc. Drawing tool is a fundamental approach that affects drawing and paint. Different tools have different properties that can create stroke and shading effects. For example, comparing pen-painted image and ink-painted image (figure 3), pen painting (figure 3 left) shows lines and strokes clearly with almost the same thickness throughout each line whereas ink painting (figure 2 right) is doing well in shading and gradient.

Figure 3 Illustration of Comparing pen-painted image and ink-painted image.

Nevertheless, ink painting by brush sometime does not only work on shading and gradient, but also work on stroke like silhouette line. Chinese calligraphy is a well-known Chinese brush technique that presents the brush effect in strokes. Also Chinese painting has unique styles created by brush. Chinese painting can be divided

into two main styles: Gongbi style and Xieyi style. Gongbi is a meticulous technique. It is usually referred to as “court-style” painting. It features on carefully detailed drawing, emphasizes the beauty of lines. It needs close attention to detail and fine brushwork (figure 4). Xieyi is a freehand technique. It is usually termed watercolor or brushwork. Water and ink is the core of Xieyi that the painter uses for exaggerating forms. Different from Gongbi, Xieyi generalizes shapes and displays rich brushwork and ink technique (figure 5).

Figure 4 and figure 5 are from http://www.absolutechinatours.com/china-travel/Traditional-Painting-Techniques.html.

Figure 4 Illustration of Gongbi style

Figure 5 Illustration of Xieyi style

Microsoft research (Baxter and Govindaraju, 2010) has done a research on modeling 3D brushes for using in realistic painting program. It explained brush stroke in mathematical term very well. The relationship of brush deformation, bend energy and friction is really useful for understanding brush effect and implementing for Computer Graphics.

Relevant research

Due to the expensiveness of CPU computation, there are many attempts to speed up the calculation including porting to GPU computation. Brown University (McGuire and Huges, 2004) presented featured-edge detection that runs entirely in GPU, and used it to create thick screen-space contours with end-cap that join adjacent thick line segments. Their technique is pretty complex to imply and it produces noise when working on tessellated meshes.

Kowalski et al. (Kowalski et al., 1999) suggested an algorithm to render 3D scene in a stylized stroke. Their algorithm is implemented to support procedural stroke-based textures, called “graftal textures”, on polygonal models. The result is to create effects fur, grass and trees. However, their technique has a problem of frame-to-

frame coherence. It has no inherent interframe consistency, which causes flickering in final rendered playback video.

Kalnins et al. (Kalnins et al., 2003) presented a methodology to render stylized silhouette path of animated 3D model. They used an ID image to determine visibility of silhouette path, and also to produce temporal coherence from one frame to the next frame. This can reduce noise and flickering during playback. However, because of the read-back of the ID image, it can slow down the frame rate.

Zakaria and Seidel (Zakaria and Seidel, 2004) presented a method to detect and render stylized silhouettes for point-sampled geometry. This kind of geometry can be obtained via 3D scanning device. Their method is firstly to project points lying on the silhouette onto color buffer, each with unique color value. Then deploy fast point-linking algorithm to form silhouette lines. This technique is really useful for supporting a new approach of modeling 3D objects. However, their work did not address the animation issue.

CHAPTER III

METHODOLOGY

In order to create stylized rendering, firstly the silhouette edges of 3D model are needed to be collected. However, there are many definitions of silhouette edges. Choudhury (Choudhury et al., 2009) has given the definitions of edges that can happen on 3D model to which this research would refer.

Type of edges on 3D model

1. Silhouette line: edge that lies against the background, which appears as the boundary of an object.
2. Self-occluding silhouette line: edge that lies against the same object portion of which is further behind.
3. Intersection line: line that marks the intersection of two 3D objects.
4. Crease line: line that defines the discontinuity of surface normal or defines the sharp edge.

Silhouette line can define an object's boundary where as self-occluding silhouette lines can define important detail of an object. These lines can be defined by the edges of 3D polygonal object, which from now on, we will call them as silhouette edges. However, they are not static edges if an object is moving or a camera is moving. Since the edges keep changing as each frame passes, they lose the continuity of the line between each frame. Moreover, finding silhouette edges on 3D objects is time consuming because it needs to compare every face in every frame.

This research will redefine 3D data structure, which would create relationship of each silhouette edge between individual frames. This also reduces repetition of calculating unnecessary edges, which results in faster rendering.

Collecting 3D data

The common file formats for storing 3D object data, which is widely used in animation industry are Wavefront OBJ and Autodesk FBX (originally created by Kaydara which later was acquired by Autodesk). Wavefront OBJ stores 3D object's components as vertices, faces, vertex normal and texture normal. All of them have only one frame data. They do not contain animation data at all. Whereas Autodesk FBX is widely used for cross-platform file transfer for 3D animated model, it only works well with

skeleton-based animation. It does not support deformation animation like blend shape and surface scaling.

In order to define polygon mesh, there are two necessary elements: vertices and faces. Each vertex would contain x, y, z coordinates in world space, and each face would contain the vertices that make the face. The vertices of each face are collected in counterclockwise direction that affect in defining face normal vector. These components' collection is similar to Wavefront OBJ, but we omit the texture coordinate and vertex/texture normal. Though each vertex may change its position as frame goes by, the relationship of face and edge are still the same. Our approach proposes to collect edge component, which would composed of two vertices and two face numbers that related to each edge (figure 6). The edge collection will help us define continuous silhouette line. The edge data will be stored as tree data structure. Also we would have array of vertex array, which would collect each vertex position for each frame.

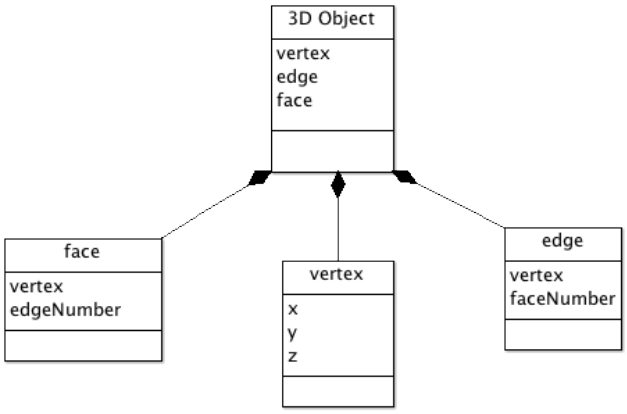

Figure 6 Illustration of Collecting 3D data as object.

Finding Silhouette edges

Firstly we need to create an array for checking edges. The size of an array is the same as the quantity of an object's edges. If necessary, we choose the pre-calculation frame by picking the frame that shows the detail of an object the most. Then we begin with the first edge, the proposed approach checks to see if this edge is silhouette edge or not. If yes, it picks a vertex from this edge and finds the connected edge for calculating silhouette edge. It also marks on the position in an array of checking edge. Before checking the new edge, we should check if the edge is already checked or not. By this method, we'll have an array of edges, which has continuous vertices. For the next frame, instead of calculating the whole edges, we calculate only

the previous silhouette edge and the edges that connect to the start and end points of each continuous silhouette edge as described in the following algorithm:

```
Calculation for silhouette edge:
For each edge in mesh:
If (edge is marked in EdgeCheck array)
Else,
    Do dot product with camera normal and each face normal in relation to current
    edge. (C•face1, C•face2)
    Do cross product with the result of previous dot products. (C•face1 x C•face2)
    If (C•face1 x C•face2  <= 0)
        //This edge is silhouette edge.
        Mark on EdgeCheck array
        Store in SilhouetteEdge array
        Pick next continuous edge.
        If (edge is not marked in EdgeCheck array)
            Do Calculation for silhouette edge
    Else
      Mark on EdgeCheck array
      End if
```

On each frame, we calculate and render the silhouette line from the EdgeCheck array and a bit more on continuous edge in relation to silhouette edge. With this approach we can create animated vector lines, which can represent a 3D object on 2D plane. The SilhouetteEdge array stores the results of silhouette edge calculation. We store this array as tree format so that we can use the tree relation as the continuity of our silhouette edge whereas the tree's branches can define the amount of silhouette lines on the scene (Figure 7). Moreover, we can use this array as a reference for calculating the next frame, which help reducing the faces for calculating silhouette edges.

On the next frame, we reset the EdgeCheck array, then do calculate the silhouette edges based on the SilhouetteEdge array of the previous frame. After finishing the loop based on SilhouetteEdge array, we pick the root and the external nodes of the new SilhouetteEdge array for finding the continuing edge and checking if it can be silhouette edge or not. With this method we can reduce the calculation on upcoming frame, and we can also get the continuous line between frames as well.

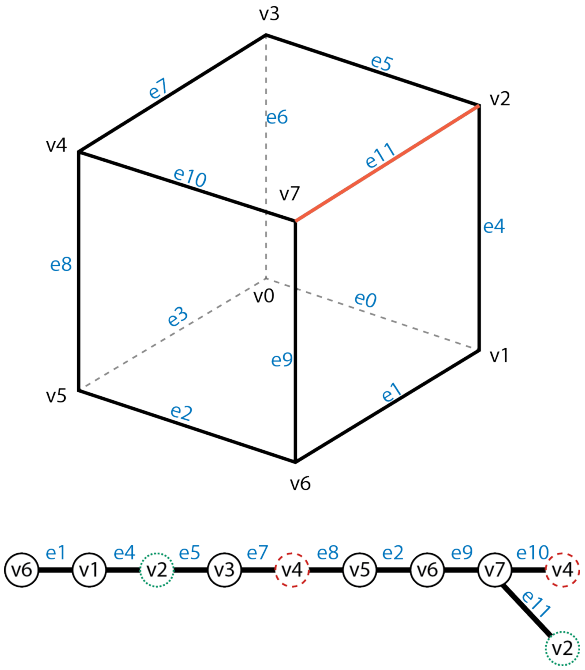

Figure 7 Illustration of Silhouette edge calculation's result.

Reducing data by using depth map

The silhouette edges that we get now are still having a problem. The silhouette edges, which should not be seen, because they are behind polygon mesh, are still showing. We need to get rid of them as well. We purpose to use depth map to check if an edge in Silhouette edges is covered by faces that are at the front. Depth map is an image that contains depth of field data. We create depth map by checking depth of field of each face in objects; however, we store only data of the nearest value. Then we compare the edge's depth of field with depth map at edge's position. If edge's depth of field value is further than the depth map, we can omit that edge by removing it from silhouette edge tree. The amount of root nodes in silhouette edge tree is the same as the amount of silhouette lines in each frame. Also the root node is used for defining the beginning of each vector line.

The first frame calculation consumes most CPU and time usage since it needs to create the starting silhouette edge tree. Depth map comparison is done after finding the starting silhouette edge tree. Then on the next frame, instead of checking every edge again, silhouette edge tree of the previous frame will be checked if each edge still appears on the current frame. At each root node and leaf node of silhouette edge tree, pick its continuing node in the initial edge tree data and do check if the newly picked edge appears on the current frame. Normally, if the new silhouette edges appear, they are the continuity of previous silhouette edges. After finishing checking for current frame silhouette edges, depth map of the current frame should be compared so that the only visible edges can exist. This technique dedicates the first frame rendering time for preparing the data and using it to reduce the redundant calculation on the next frame.

Creating Chinese brush style

Due to the limitation of OpenGL, normal shader cannot be applied directly to the lines. Lines should be implemented as polygonal quad strips, and then it can be applied by textured mapping. The texture map is an image of stroke tip.

Figure 8 Illustration of Stroke tip image.

Converting line to polygonal quad strip can be done by firstly finding vectors that are perpendicular to an edge in silhouette edge tree at the starting vertex and ending vertex. There should be two vectors of opposite directions at each vertex. Then set the length of each vector as half of the desired line thickness value. At this state, the four vectors can be implied as the four new vertices. The four new vertices of each visible silhouette edge will be used for defining quad strips in OpenGL

After getting quad strips of each visible silhouette line, apply the shader with stroke tip image to create the brush like look. The stroke tip image (figure 8) is a gray scale image of brush tip shape. It will work as alpha channel of the shader so that the color of the line can be define later.

CHAPTER IV

RESEARCH RESULT

This research is experimented in Objective-C based OpenGL. It is run on 2.66 GHz Intel Core i7 with 4Gb memory. The test run in the beginning is very slow due to the amount of data collection. It is recommended to start testing research's renderer with fundamental geometry like torus. Torus is chosen, because its shape is not too simple. Also it has self-including silhouette edges, which are the main edges that would be problem when finding silhouette. The result does not perform well with Chinese brush shader, It does not look like brush stroke as expected (figure 9).

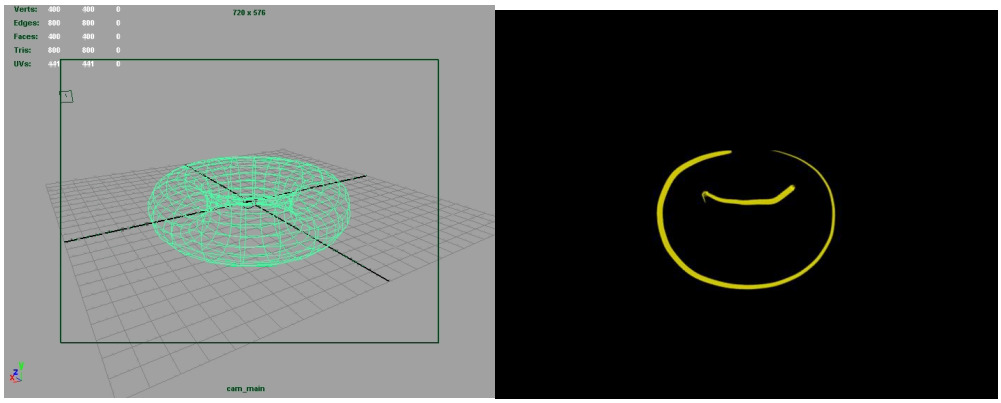

Figure 9 Illustration of Torus rendering.

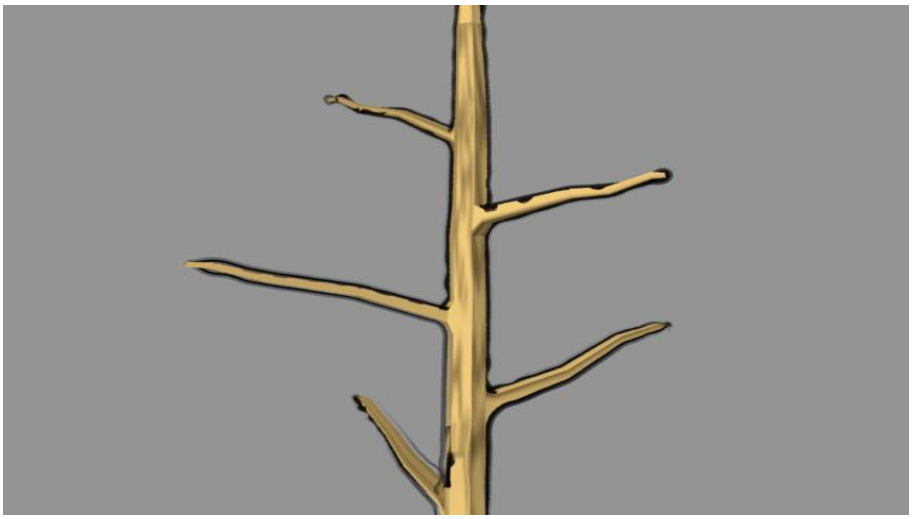

Figure 10 Illustration of Tree branch rendering with watercolor surface shader.

Surface shader has been attempted (figure 10) with expectation of helping the line stylization look more brush alike, but it only gets better a little bit. When comparing to genuine Chinese brush painting (figure 11), it looks much different.

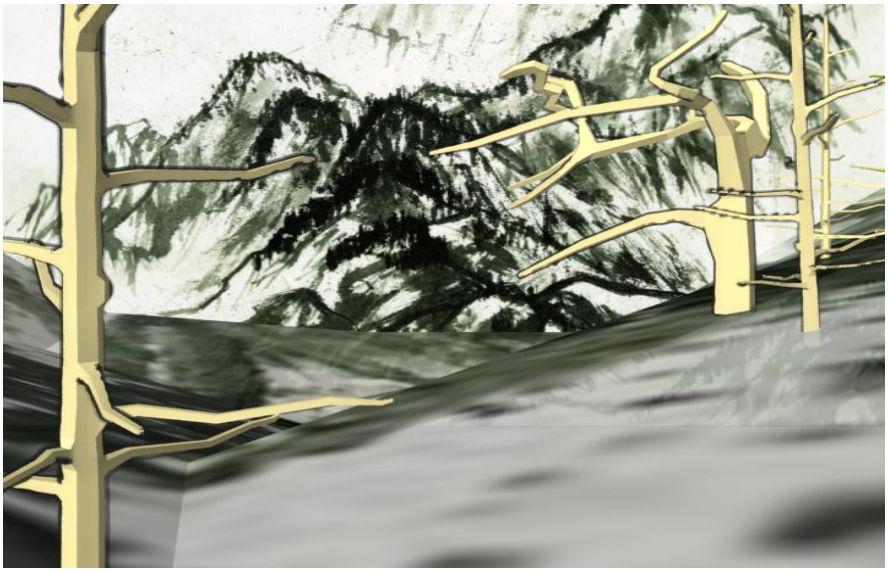

Figure 11 Illustration of Scene rendering with background of Chinese brush painting.

The final result still looks cartoony, because of the color theme and the surface shader. However, the speed of rendering is working well since there is less repetition of edge calculation. Without redundant deduction technique, it could take more than 3 hours to calculate 100 frames of silhouette line of three-dimensional rotating torus. However, with presented technique, it takes less than 2 hours to calculate the same animated torus.

Unfortunately, there is no consistency of line's starting point between each frame. This event often happens with silhouette line that has loop property of which the starting point is the same one as its ending point. Therefore, when doing playback, the loop lines perform jittering (figure 12).

Also there is a question about the result of the rendered image that if it is really similar to the physical Chinese brush stroke. The uncomplicated methodology to compare is suggested by Cole et al. by superimposing images over each other and use difference filter to see the compound area. Testing with circular stroke (figure 13), the left image is the computer rendered stroke of 3D sphere. The middle image is Zen's circle calligraphy by Kanjuro Shibata. It is done by using ink and big round tip brush. The right image is the comparison of previously mentioned images by using difference

filter. The yellow area shares the space that both strokes are taking. The blue area is the space that computer generated stroke is taking, but the physical stroke is not. The black area is where the physical stroke is taking, but the computer generated one is not. When merging the black area and the blue area, it seems that both images overlap almost 75% as recommended in Cole et al.'s research.

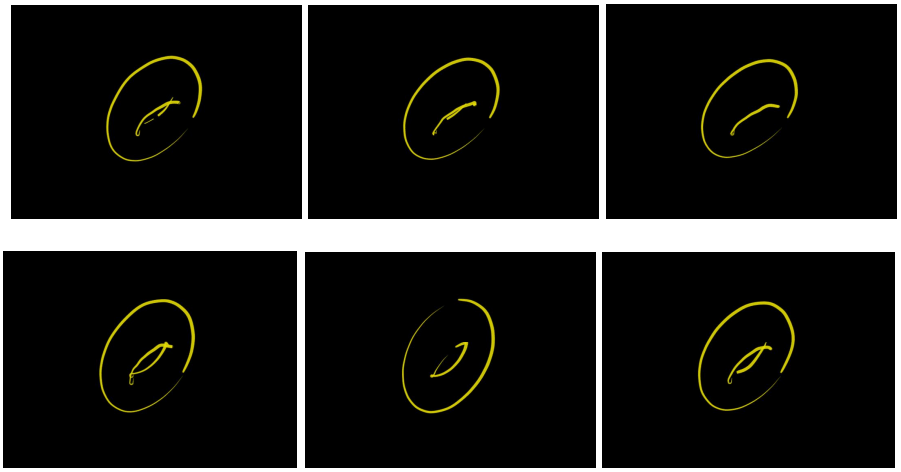

Figure 12 Illustration of Animating torus.

Figure 13 Illustration of Comparing computer generated stroke and physical stroke.

CHAPTER V

CONCLUSION AND DISCUSSIONS

General Conclusion

This thesis presents the technique to manage 3D data for detecting silhouette edges on 3D animated model. Then it uses that data for finding silhouetted edges and creating Chinese brush stylization. The result in time-consuming aspect is very satisfied. By managing 3D data, it helps reducing calculation repetition. Moreover, it also creates the continuity of line when rendering as sequence. However, the result in brush shading aspect is still having a problem. It still lacks some Chinese brush painting properties like bristle effect. Also it is because of OpenGL limitation for stylizing line stroke. It would be better if there is a procedural shader that supports vector line rendering.

Discussions

Though the results do not look like Chinese brush stroke as expected due to OpenGL limitation, it can still create brush alike stylization. There are also some disadvantages of presenting calculation reduction. If the starting frame does not show every silhouette line that should appear in some frame of animation then that line is lost forever. Also if some silhouette lines are missing as frame goes by, they will never happen again after that frame. There should be a method to give weight of silhouette line that can be used for determining re-checking after its invisibility.

Moreover, there's still a problem of jittering image during sequence. It is the problem of line consistency between frame which occurs in many researches. However, in this thesis, the problem of line consistency often happens with the loop lines. There is still no certain methodology to explain which point of loop line should be the starting point. Even though having artists drawing the same loop stroke, different artists do the different starting point. However, if setting some rule like the starting line should be related to something fixed in the scene, the starting points are almost the same. Therefore, light source should be able to guide the starting point since some artists tend to have stroke pressure and thickness depended on light intensity.

REFERENCES

Goral, C.M., Torrance, K.E., Greenberg, D.P., Battaile, B. Modeling the Interaction of Light Between Diffuse Surfaces: Proceedings of the 11th annual conference on Computer graphics an interactive techniques (SIGGRAPH'84). pp. 213-222, ACM New York, USA., 1984.

Greenberg, D.P., Cohen, M.F., and Torrance K.E. Radiosity: A method for computing global illumination. The Visual Computer (1986): 291-297.

Jensen, H.W. Global Illumination using Photo Maps: Proceedings of the 7th Eurographics Workshops on Rendering. pp. 21-30, 1996

Meier, B.J. Painterly Rendering for Animation: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). pp. 477-484, ACM New York, USA., 1996

Cole et al. Where Do People Draw Lines?: Proceedings of ACM SIGGRAPH '08, ACM New York, USA., 2008

Gooch, A., Gooch, B. Shirley, P., Cohen, E. A Non-Photorealistic Lighting Model For Automatic Technical Illustration: ACM Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH'98). pp. 477-452, 1998.

Northrup, J., Markosian, L. Artistic Silhouettes: A Hybrid Approach: ACM Proceedings of the 1st international symposium on Non-photorealistic animation and rendering (NPAR '00). pp. 31-37, 2000.

Baxter, W., Govindaraju, N. Simple Data-Driven Modeling of Brushes: ACM Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (I3D'10). pp. 135-142, 2010.

McGuire, M. and Hughes, J.F. Hardware-Determined Feature Edges: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering (NPAR '04). pp. 35-47, 2004.

Kowalski, M., et al. Art-Based Rendering of Fur, Grass, and Trees: ACM Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH'99). pp. 433-438, 1999.

Kalnins, R., Davidson, P., Markosian, L., Finkelstein, A. Coherent Stylized Silhouettes: ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003, Volume 22 Issue 3, July 2003. pp. 856-861, 2003.

Zakaria, N., and Seidel, H.P. Interactive Stylized Silhouette for Point-Sampled Geometry: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE '04). pp. 242-249, 2004.

Choudhury, A.N.M.I. and Parker, S.G. Ray Tracing NPR-Style Feature Line: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering. pp. 5-14, ACM New York, USA., 2009.

APPENDIX

LIST OF PUBLICATIONS

Part of this work is published in the following articles.

International Conference Proceedings

Nucharee Thongthungwong and Rajalida Lipikorn, “Data Reduction for Finding Silhouette edges on 3D-Animated Model”, 5th International Conference on Future Computer and Communication (ICFCC2013), Phuket, Thailand, May 26-27, 2013.

VITAE

Nucharee Thongthungwong was born in Bangkok, Thailand, on 28th November, 1977. She received her Bachelor degree of Architecture major in Industrial Design from King Mongkut Institute of Technology Ladkrabang in 2002. She had worked as graphic designer for A26 Advertising Company for a year, then moved to work with Dida Video Production as 3D Motion Capture and animator for 2 years. Currently, she is working on her master degree in image rendering at Computer Science and Information Technology Program, Department of Mathematics and Computer Science Faculty of Science, Chulalongkorn University.