DEVELOPING GENERALIZED CROSS HATCHING SHADER APPROACH FOR NON-PHOTOREALISTIC RENDERING

A Thesis

by

YUXIAO DU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Ergun Akleman
Committee Members,	Richard R. Davison
	Rodney Hill
Head of Department,	Tim McLaughlin

August 2017

Major Subject: Visualization

ABSTRACT

In this research, I present a method for rendering a geometric scene that has the look and feel of artistic hand drawings, particularly using a medium such as charcoal or cross-hatching. While there have been many approaches to non-photorealistic (NPR) renderings in the past two decades, there seems to be very little research done on how to obtain such charcoal or cross-hatching effects, especially with attention to reflections and specularity, which often at times seems to break the illusion of the drawing effect.

I developed a new class of techniques, using a Barycentric shading method, that allows the non-photorealistic rendering of a variety of artistic drawing styles. My approach can be summarized as follows: (1) a Barycentric shader that can provide generalized cross-hatching with opaque multi-textures, (2) a Barycentric shader using transparent multi-textures, and (3) a texture synthesis method that can automatically produce crosshatching textures from any given image.

DEDICATION

To my mother, my father, and my grandmother.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Ergun Akleman and Professor Richard Davison of the Department of Visualization and Professor Rodney Hill of the Department of Architecture.

The methodology and results in chapter 3 were conducted by the student and Professor Ergun Akleman and published as a poster in 2016 in ACM SIGGRAPH conference titled "Charcoal Rendering and Shading With Reflections".

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study has been partially supported by the Dreamworks Animation Scholarship and the Pixar Aggies scholarsip.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

1.1 Motivation and Inspiration

With the current and innovative techniques in computer graphics, where motion pictures are able to produce ever more realistic renders of fantastic worlds, it is hard to imagine that producing non photorealistic images is still an ongoing challenge for professionals in the field of computer graphics. During the last two decades many non-photorealistic (NPR) rendering methods are developed to simulate charcoal drawing [7] [8]. However, shader development is still a very active area of research, and professionals within the industry is constantly trying to find new ways for control of the tools to create the desired artistic style. No work has been done to create global illumination effects such as reflection into charcoal styled renderings. This is most likely due to the fact that traditional charcoal drawings do not commonly include reflections. Furthermore, with current approaches to non-photoreaslitc rendering, artists do not yet have the ability to create a completely new cross-hatching style. All current works rely on methods that have only been seen using traditional graphite techniques. However, my approach will show that it is possible to analyze such examples to include global illumination effects into charcoal drawing as well as create new and exciting cross-hatching styles for the artists' own choosing.[9]

1.2 Introduction

After decades of advancement in computer graphics, where a primary goal is to let viewers not realize that computer generated graphics is even being used, why are artist and professionals looking for ways to revert back to producing non-photorealistic images? Grabli and Yan addresses this by noting that it is for the most part a matter of artistic style. [10][11] Yan even describes NPR as artistic rendering, and one of the natural ways to generate artistic rendering would be to imitate the painting style of an artist. However,

this proves to be challenging and continues to be a problem, as artists stroke patterns and colors can vary greatly to convey their emotion [11].

In addition to artistic style, NPR can be applied in technical areas such as the medical field [12] and technical manuals, where contour lines and clear edges are desired for more descriptive purposes, allowing for an educational environment. One of the primary applications for a programmable NPR system is technical illustrations, where occlusion property is used to focus on individual elements of a drawing of an engine [10].

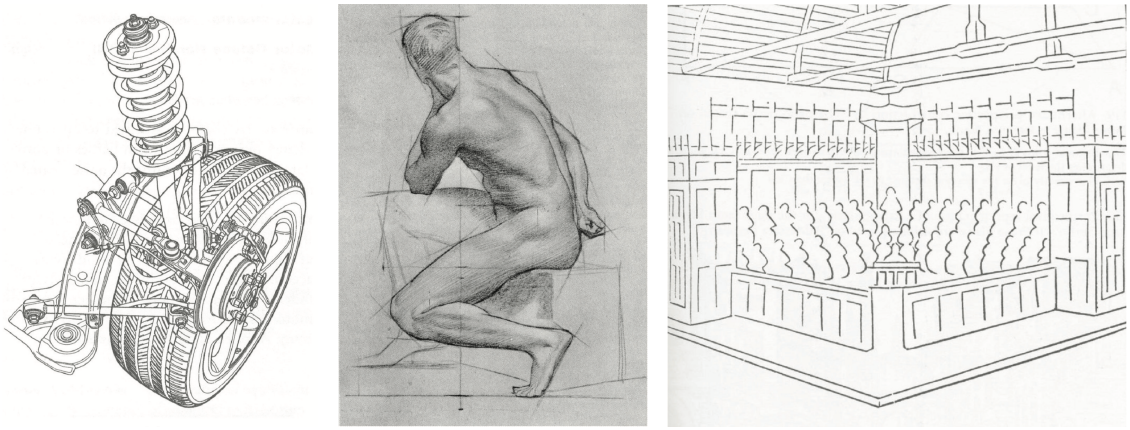


Figure 1.1: Artistic examples including technical illustrations to emphasize a subset of the assembly elements and charcoal drawing where non-photorealistic rendering could be applied.

My goal for this research is to create an approach that can easily be applied to standard pipelines, which will achieve and mimic the effects of artistic styles such as charcoal and cross-hatching. While past approaches to reach this desired result has been time-consuming and difficult for artists to understand, I am developing a Barycentric shader that requires minimal amount of work for the desired result. The render should have the artistic quality of hand-drawn artworks as well as give the believability to the viewer that they are seeing a drawing instead of a rendering of 3D objects. Creation of this generalized

cross-hatching shader is driven by the following goals:

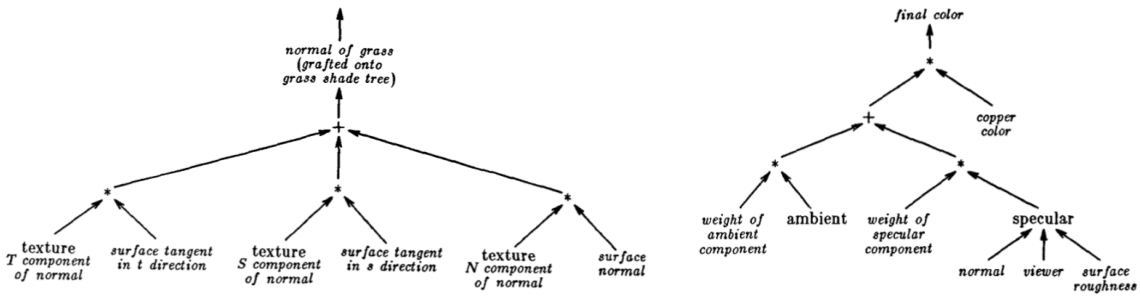
- Create 3D models that traditional artists would draw. The models would be still life or detailed human figures often seen in traditional drawings.
- Create multiple opaque textures of variant shades using charcoal on paper. Each texture would need to be fairly consistent in texture, stroke width, and value throughout paper on which it is drawn. I would be using this texture as part of my method in shader development.
- Create multiple transparent textures using Adobe Photoshop. Each transparent texture would show very simple line strokes that could easily overlap each other with simple rotations of the texture.
- Create an algorithm for image synthesis. The algorithm would greatly help with the consistency of the textures, where the gaps of white or dark areas would as a whole look more uniform throughout the texture image.

2. BACKGROUND AND LITERATURE REVIEW

Many works have been done in the past to attempt the effect hand-drawn styles in rendering. In this chapter, I will examine the works of my predecessors related to my research of non-photorealistic rendering.

2.1 Shading Trees and Shading Language

From developing games to animated films, the development of the shaders and shading frameworks has always been a very important task. Shade Trees architecture (See Figure 2.1) and shading languages laid the foundations of the procedural shader concept to create a desired look-and-feel [1, 2]. Cook's conceptual development of the shade tree allowed artists immense control of the shader, manipulating light source specifications, reflections, and atmospheric effects independently [1]. The artist was able to selectively control every aspect of the shading process, allowing for more flexibility.

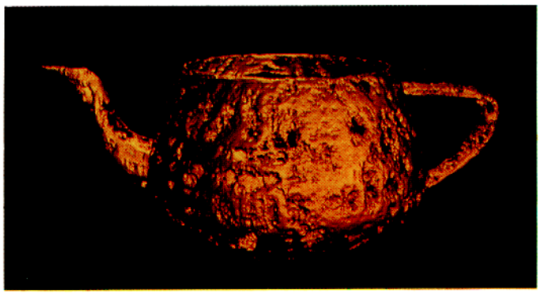


(a) normal (b) final color

Figure 2.1: Conceptual examples of shade trees for grass [1].

The Shade Tree concept was then utilized and adapted by Perlin's image synthesizer and Hanrahan's Renderman shading language, where both introduced full shader lan-

guages for programming shading computations for rendering systems [13, 2]. Carrying Cook's Shade Tree further, Perlin developed a programming language with a full set of arithmetic and logical operators [13]. However, Cook's distinction of the shading process was not utilized by Perlin. He instead created a "pixel stream" model, where the shading was done in post-processing. Utilizing the methods proposed by both Cook and Perlin, Hanrahan created a "high-level language" (See Figure 2.2), which he describes as very "easy to use", allowing "shading calculations to be expressed naturally and succinctly" that included a well defined user interface between the shading modules and rendering program[2]. Despite the success of the shader and shader development, however, obtaining a desired style continues to be a challenge.



```
surface
dent( float Ks=.4, Kd=.5, Ka=.1, roughness=.25, dent=.4 )
{
    float turbulence;
    point Nf, V;
    float i, freq;

    /* Transform to solid texture coordinate system */
    V = transform("shader",P);

    /* Sum 6 "octaves" of noise to form turbulence */
    turbulence = 0; freq = 1.0;
    for( i=0; i<6; i+= 1 ) {
          turbulence += 1/freq * abs( 0.5 - noise( 4*freq*V ) );
          freq *= 2;
    }

    /* Sharpen turbulence */
    turbulence *= turbulence * turbulence;
    turbulence *= dent;

    /* Displace surface and compute normal */
    P -= turbulence * normalize(N);
    Nf = faceforward( normalize(calculatenormal(P)), I );
    V = normalize(-I);

    /* Perform shading calculation */
    Oi = 1 - smoothstep( 0.03, 0.05, turbulence );
    Ci = Oi * Cs * (Ka*ambient() + Ks*specular(Nf,V,roughness));
}
```

Figure 2.2: Procedural texture on coroded teapot using Hanrahan's shading language, now known to be the industry standard Renderman shading language [2].

2.2 Understanding Drawing Principles for Non-Photorealistic Rendering

While developers are making progress in finding better ways to produce desired renderings of 3D models, the challenges of non-photorealistic rendering (NPR) are great and many. Producing rendering that give the desired look and feel of something hand-drawn is extremely difficult. Highly qualified lighting and technical directors are still wondering how to make artistically stylized shaders without it looking and feeling like the result has been artificially generated through a rendering system.

To start off, "where do people draw lines"? This question has been asked by most if not all researchers prior to their approach of non-photorealistic rendering, specifically in contouring [12, 10, 4]. We can see an artist's process in making a pencil drawing on paper (See Figure 2.3), but how and why does he make the decision to place the marks there? It is usually assumed that style attributes (i.e. stroke thickness or line omission) are not randomly chosen by artists. Grabli believes that the nature of a line is dependent on the distance to the viewer, but includes that this assumption was never explicitly exploited nor articulated [10].

Cole presents a study (See Figure 2.3) where twenty nine people consisting of art students and professional artists were asked to produce line drawings based from twelve given images of 3D shapes in a controlled environment [4]. The study was designed in such a way where the drawings would be analyzed using computer line definition to show how the locations of each artist′s line compares to that of another artist, finding relationships between line location of artists′ drawings and surface geometry, lighting, and viewing conditions [4]. The participant were given specific instructions such as to refrain from drawing lines that may represent shadows as their purpose was to convey the shape of the 3D objects. To add precision without the trade off of artistic decision, participants were asked to first make a free- hand drawing, and then to trace over a light version of the 3D

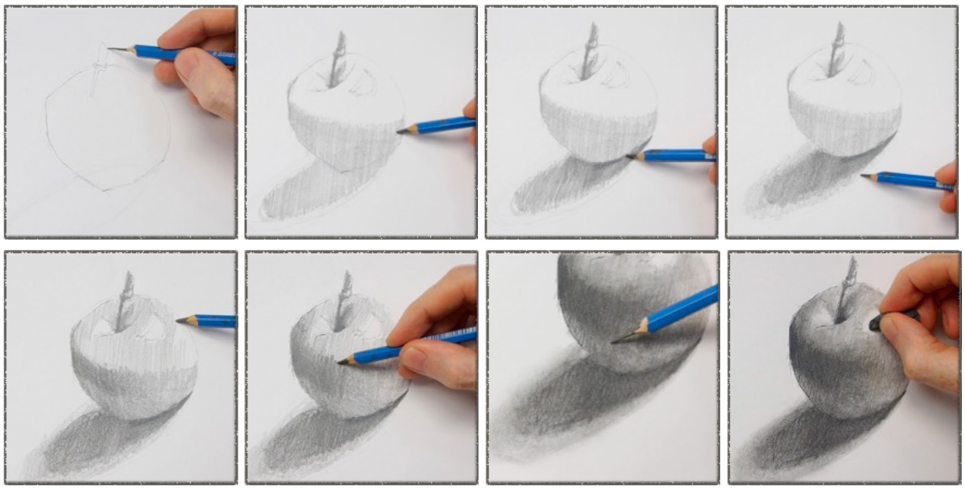


Figure 2.3: General Drawing Process, where an artist draws an apple freehand from beginning to finish [3]

image repeating the same lines they drew from their free-hand drawing. The results shows that 75 percent of overlapping lines from one artist to another are from occluding contours of the geometric object, while most of other lines seemed to represent large gradients [4]. However, Cole′s result only covers one drawing style, and he admits that his study pool is small. A more extensive study with a larger pool of participants is desired.

2.3 Related Works

Barycentric shader is a new approach introduced by Akleman for simplifying shader development in an intuitive and streamlined process. Barycentric shaders can be understood with the following equation:

$$C(u,v) = \sum_{i=0}^{M} B_i(t) T_i(u,v)$$

where $C(u,v)$ is rendered color of the point (u,v), $T_i(u,v)$'s are control images and t is one of the shading parameters such as diffuse parameter, specular parameter, ambient oc-

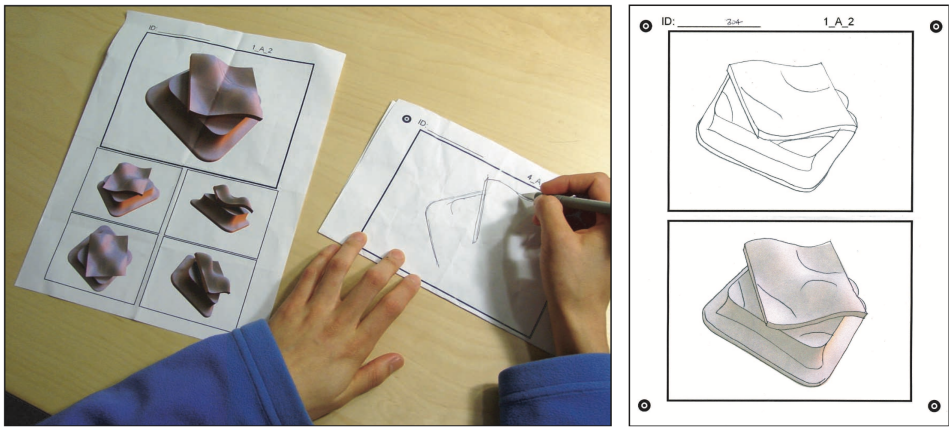


Figure 2.4: Artist makes free-hand drawings from a prompt page (left)[4].

clusion or shadow and $B_i(t)$'s are basis functions that satisfy partition of the unity property [5]. The consistency of the results can be enforced regardless of the number of shading parameters computed during rendering by using basis functions that satisfy the partition of unity [5]. Desired styles are obtained from key decisions regarding utilizing control images and basis functions (See Figure 2.5).

Figure 2.5: Using control and weight images to satisfy partition of unity, where the final image (right) created by taking the weighted average of the control images (left) [5].

Using the Barycentric shading method, any standard rendering pipeline along with

global illumination can incorporate these shaders into a streamlined process. Recently, the method has been used to successfully emulate existing artists' styles[6].

Figure 2.6: Chinese Painted 3D scene using the Barycentric shading method [6]

Liu is able to create a Chinese Painted 3D scene with reflections on based on a painting by Yang Ming-Yi, a contemporary Chinese painter(See Figure 2.6). By studying the paintings of the artist, Liu has identified crucial elements for the paintings: (1) Value and Tone, (2) Shape and Form, (3) Water Reflection, and (4) Layout and Composition [6]. Liu addresses that the reflection of the water in the rendered scene resulted from using weighted images [5], a combination of light and dark tone reflection images using a reflection mask and fade-out parameter [6]. The final image is created as a result of several rendered layers, each with a slighting different camera position and then later composited in post-process.

While the the Barycentric method proves to be extremely promising with successful results, very little work has been done to showcase the method without post-processing.

Liu's Chinese painting rendering is the result of multiple layers of renderings that has been composited together, which can be a tedious task in itself. Furthermore, the capabilities of Barycentric shaders can be expanded to create more styles, even non-existing ones. In the following chapters, I will propose a method to create generalized crosshatching effects to expand on the Barycentric shader. In addition, I will introduce an image-synthesis algorithm, which generates control images for non-existing styles.

3. BARYCENTRIC SHADING WITH OPAQUE MULTI-TEXTURES

In this chapter, I will present a way to create a charcoal shader using Akleman's Barycentric shading concepts [5] using hand-made charcoal textures as control images. As an extension to Barycentric shader, I will explain the relevance of using a degree 0 b-spline[1] for creating a charcoal drawing effect.

Finally, I will introduce the artist drawing process, which I call the artist hierarchy, and how following the same process within the shader will generate the desired charcoal style in the final render.

3.1 Shader Development

As addressed in the previous chapter, a Barycentric shader is a shading framework using Barycentric algebra. Using a basis function, the shader allows for very intuitive art-directed control [5]. It utilizes a series of control images as weight images, and it can be summarized using the equation

$$C(u,v) = \sum_{i=0}^{M} B_i(t) T_i(u,v)$$

where $C(u,v)$ is the rendered color of the point (u,v), $T(u,v)$ is the texture image, M indicates the number of parameters used, and t denotes illumination, such that t=1 means fully illuminated and t=0 means not illuminated.

$$\sum_{i=0}^{M} B_i(t) = 1 \qquad and \qquad B_i(t) >= 0$$

$B(t)$ is the basis function that satisfy partition of the unity property. The summation

[1] Y. Du and E. Akleman, "Charcoal rendering and shading with reflections," in ACMSIGGRAPH 2016 Posters, p. 32, ACM, 2016.

of the weighting coefficients $B(i)$ must be equal to 1, where $B(t)$ is equal or greater than 0.

3.1.1 0 Degree B-spline

For my charcoal shader, I am using a zero-degree b-spline function [9] for the weights $Bi(t)$. It is a simple function that is equal to one when parameter t is within an interval. Otherwise, it is equal to 0. This guarantees that a clean texture that provides approximately the desired color value is obtained. Note the following degree zero B-spline

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

where $N(t)$ is the zero degree B-spline basis function that is equal to 1 when parameter t is within an interval, and 0 otherwise. Using my choice of control images, I can mix them along the 0 degree B-spline.

To control style, the most important component is the choice of control images. All control images are brought into shader development. The control images used in this method are nine hand-drawn charcoal textures using the charcoal on newsprint paper (See Figure 3.4). Each texture substitutes as a tonal value of the charcoal drawing, which will be then applied to the 3D geometric mesh. The textures are scanned and imported into an image-editing software to make the textures repeatable. Finally, the control images have been adjusted in brightness so that average brightness difference between textures are consistent. The idea is to create a consistent brightness ramp often seen in drawing practices (See Figure 3.4).

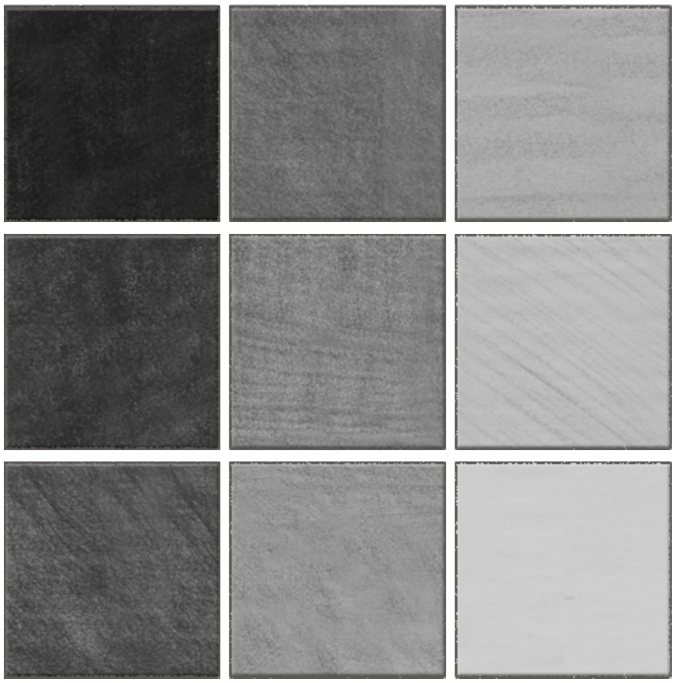


Figure 3.1: Set of hand-drawn opaque charcoal textures used as control images for Barycentric shader

3.1.2 Artist Hierarchy

Consider the artistic process of drawing an apple mentioned in the previous chapter (See Figure 3.3). Breaking down the process into steps, the artist starts with an outline. The artist then starts adding shade to the apple with some basic shading to give the form volume. Shadows are inserted before the artist goes back in to the crevices to add some more darkness. Those dark regions in crevices is often rendering as ambient occlusion in the computer graphics world. The artist then redefines the edges and outlines the apple in areas once again. Reflection is then added to the apple before the artist finally uses an eraser to give the apple specularity. The work-flow can be viewed as an artist's hierarchy, layering steps on top of each other.

With all of the steps in the artist's hierarchy, it would be extremely difficult to incor-

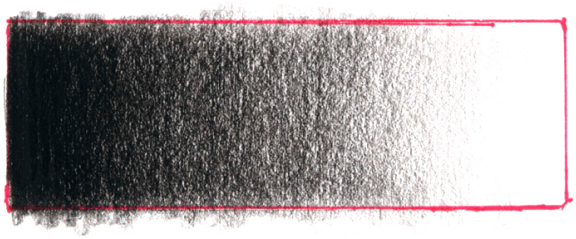


Figure 3.2: Hand-drawn charcoal gradient ramp

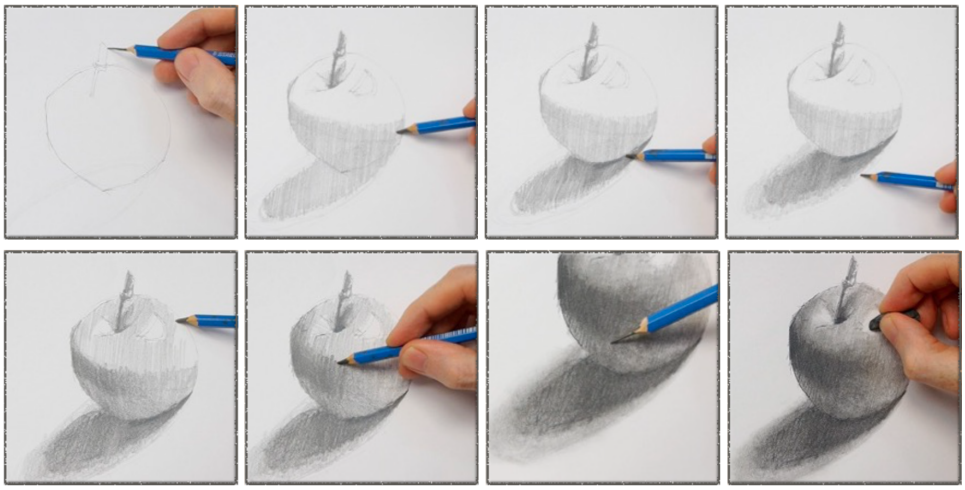


Figure 3.3: Artist Hierarchy, the general drawing process and work-flow of an artist. The steps can be see as follows: outline + base + shadow + ambient occlusion + outline + reflection + specular.

porate all of the parameters and keep track at the same time using a generic parametric function. As a solution, the shader can be simplified by creating a hierarchical system that emulates the artist's hierarchy. By making each step of the process as a shading parameter, the proposed Barycentric shader uses the following artist's drawing order:

$$Outline + Base + Shadow + AmbientOcclusion + Outline + Reflection + Specular$$

Compare that with the hierarchical structure used by the charcoal style Barycentric

shader:

$$Diffuse + Shadow + AmbientOcclusion + Outline + Reflection + Specular$$

Using the initial B-spline formula $C(u,v) = \sum_{i=0}^{M} B_i(t)T_i(u,v)$, the shader regards diffuse as t_0, shadow as t_1, ambient occlusion as t_2, outline as t_3, reflection as t_4, and specular highlight as t_5. Combining these parameters results in the final rendered image (See Figure 3.4).

Figure 3.4: Charcoal style rendered image by combining shading parameters in the same order as the artist hierarchy.

This method reduces the complexity of shading by using a hierarchy similar to the

drawing order of an artist as shading parameters. For specular highlights, a simple paper texture or white chalk texture can be used.

3.2 Texture Mapping

Three primary types of texture mapping methods are considered for the charcoal Barycentric shader: UV mapping, camera projection, and tri-planar projections. Let us review the three projection methods (See Figure 3.5).

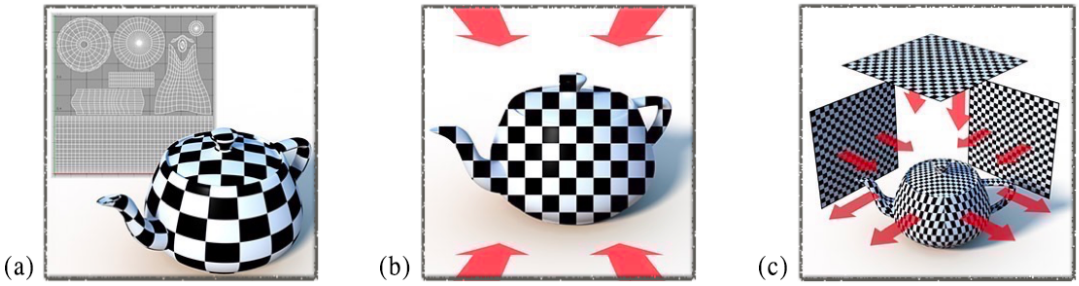


Figure 3.5: Texture Mapping methods: (a) UV Mapping, (b) Camera Projection, (c) Tri-Planar Projection

A model have different sets of UV coordinates, which are referred to as UV sets or Map channels. The UV mapping method uses the model's UV coordinates that are assigned to the model's vertices. The model can be unwrapped like the wrapping paper to an object, laying the paper flat for texturing. This is a very common practice for texture artists to locate the surfaces of the model for texture placement. One of the advantages of the UV mapping method is that it is seamless (e.g. no seams when the texture is wrapped around an object). Moreover, there is very little distortion of texture around the edges of a 3D model being textured. In addition, UV mapping follows the curvature of the form of the 3D model being textured. This feature is desirable for the purposes of the charcoal shader, because the strokes of the charcoal lines should follow the curvature of the form that is

being drawn. However, mapped textures must be specific to the 3D model being textured due to the nature of the texture being placed on the specific model that had been UV-unwrapped. This dramatically limits the utility of the UV-mapping method. Furthermore, because the textures must be specifically generated for a shape, the UV mapping method is also very time-consuming.

Now let us take a look at Camera Projection Mapping. This method projects an image onto the 3D model from the camera. It creates an illusion of detail for the object, when in reality the textures are not actually set on the model itself. This method is commonly used by matte painters in the film and television industries. Similar to the UV mapping method, the camera projection method can also produce a seamless texture for a 3D model. Unlike the UV mapping method however, the model does not need to be UV unwrapped, so there is no need for extensive human labor to produce the texture. This means that the textures are not model-specific, allowing the textures to fit on any model that the artist deems appropriate. Unfortunately, as promising as this method seems initially, this method has only been able to be successful for still images. When tested for animation, due to the texture positions reliant on the camera position, such a limitation would generates a swimming texture effect during animation. As the model moves and rotates, the textures do not follow the model, causing the textures to appear separated from the geometry and gives a very bizarre look and feel when rendered (See Figure 3.6).

Due to the uv mapping method being model specific and time consuming and the camera projection method giving swimming texture effects, another method is necessary to combat the disadvantages. The final texture mapping method to be considered is the tri-planar projection method. In this method, a texture is mapped three times with planar maps along the X, Y, and Z axes. These maps are then blended based on the angle of the face. Like UV mapping, there is very little distortion near edges of a texture. Similarly like the camera projection method, there is no need for human labor. Moreover, the tri-planar

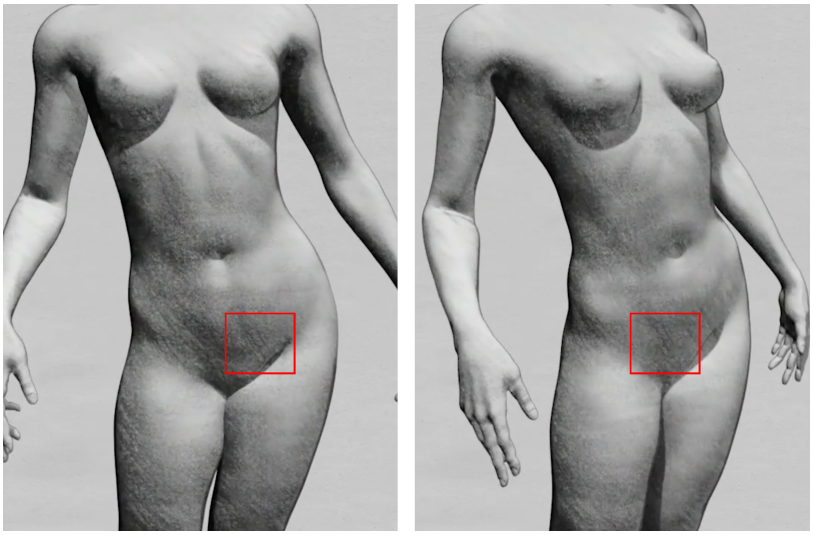


Figure 3.6: Two frames of animation using camera projection: As the model rotates, the texture can be seen as not following the rotation, causing a swimming texture effect

projection method will create textures that can follow the curvature of a 3D model, which is often an artistic decision in crosshatching lines.

	UV Mapping	Camera Projection	**Tri-Planar Projection**
Pro	• Seamless • Very little distaortion of texture around edges	• Seamless • No distortion of texture around edges • No need for human labor • Maps are not model-specific	• **Very little distortion near edges** • **No need for human labor** • **Maps are not model-specific**
Con	• Mapped textures cannot be used universally (model specific) • Time consuming	• Swimming textures (ugly for animation)	

Figure 3.7: Pros and Cons for UV Mapping, Camera Projection, and Tri-Planar Projection

After reviewing the advantages and disadvantages of the three methods, tri-planar projection proved to be the most appropriate texture mapping method for this project (See

Figure 3.7).

3.3 Shader Implementation

Implementation of the charcoal Barycentric shader is used using a visual programming language. The 3D software used is Autodesk Maya for its sheer power and versatility. Its node editor system allows the connecting components of any given data without having to go through walls of code. I have chosen to use Mental Ray as my renderer of choice, where surface luminance is given as a node in the node editor within Maya (See Figure 3.8), providing light information for users to have full control over the effect of light on the diffuse surface.

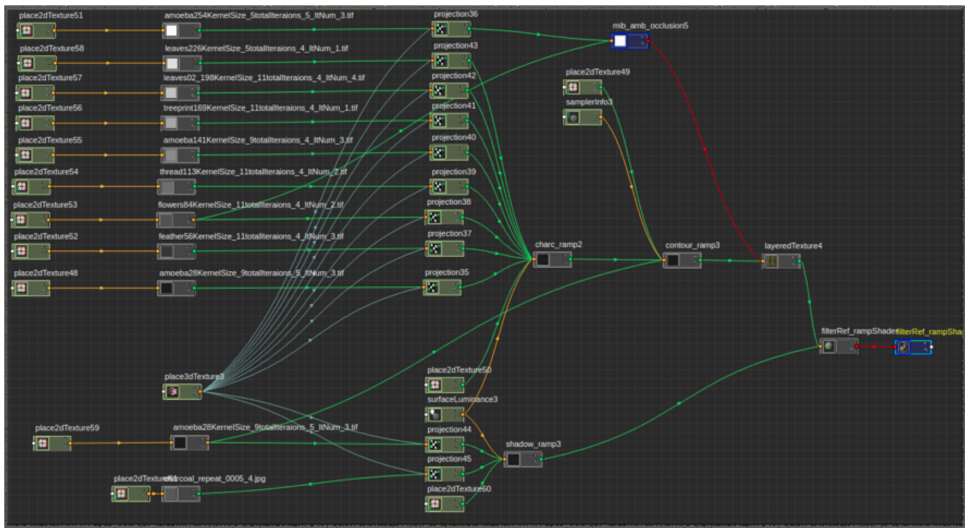


Figure 3.8: Creating the Barycentric shader using visual programming language

To start off, I have imported objects into the scene. The models used in my examples are nude bodies, apples and teapot, and vases. I have chosen these models specifically, because such objects are common content for actual charcoal drawings. However, any model could potentially be used for the scene, as the shader can be universally on any

mesh .

Within the scene, there are three components: a background plane, the mesh model, and an area light. The background plane's purpose is to give a constant texture, which is the paper texture. It will give the illusion that the final rendered model a drawing of an object on paper (See Figure 3.11). However, if user plans to incorporate shadows cast by the model, the shader will be applied to the background plane as well (See Figure 3.12). Similar to setting up a still-life scene for drawing, the area light is placed to illuminate the scene with the 3D model in however the user sees fit (See Figure 3.9).

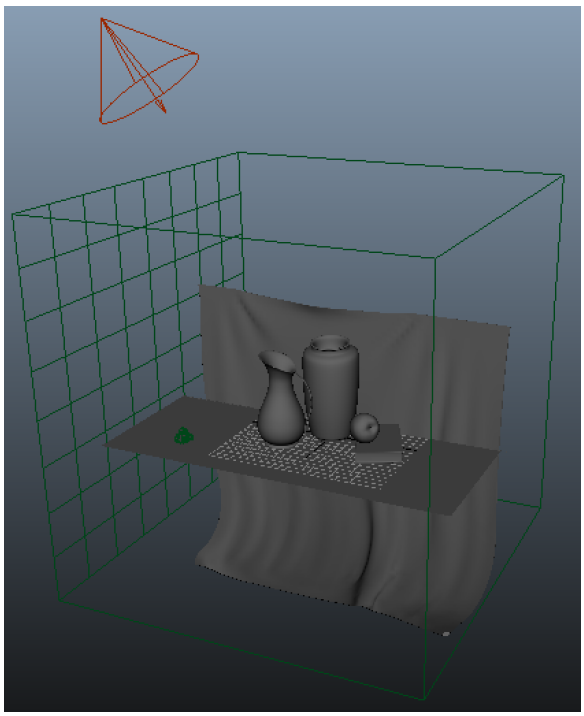


Figure 3.9: Vase scene creation in Autodesk Maya

The key factor to making the shader Barycentric is the ramp node, which allows the interpolation of the control images using Barycentric functions. Using the ramp node, I am

able to take the out-color of the control images and visually insert them along a horizontal slider (See Figure 3.10) from the darkest control image to the brightest. This ramp can be used for all parameters (i.e. diffuse, outline, shadows, ambient occlusion, reflection, and specularity) of the shader, following the artist's hierarchical structure. A surface luminance node is used, where the out-color of the the surface luminance is plugged into the ramp's UV coordinates to ensure the texture used by the ramp is based on the light value. This will allow the shader to use brighter textures in areas with more light and darker textures in areas with less light within the chosen set parameters of the ramp. The interpolation function of the ramp (in this case the zero-degree b-spline) is also set within the ramp node. In other words, depending on the brightness of the surface of the model from the light within the scene, the shader will change the texture from the selection of control images within the shader.

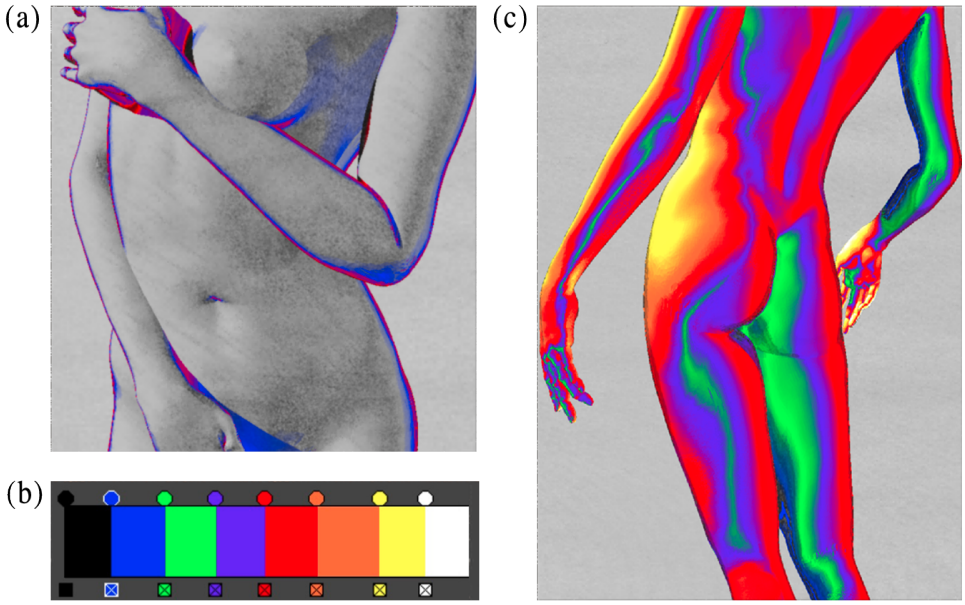


Figure 3.10: Visually mapping parameters onto the model using the ramp node. (a)Using a rainbow ramp to visually see the contours. (b)Ramp node with each color substituting a control image. (c)Visually see which control image is used on which areas of the model

Projection and place-3D-texture nodes are used for tri-planar projection. The projection node allows the user to choose the type of projection method, while place-3D-texture allows the user to visually see the projection planes on the interface. A projection node is attached to each of the nine control images for the user to have full control of each control image's mapping method. In the charcoal shader's case, only tri-planar projection is used.

3.4 Results

The following are the rendered results using the opaque control images for the Barycentric charcoal shader.

Figure 3.11: Figure Drawing Charcoal Rendering

Figure 3.12: Vase Charcoal Rendering

Figure 3.13: Teapot Charcoal Rendering

4. BARYCENTRIC SHADING WITH TRANSPARENT MULTI-TEXTURES

Let us review the drawing process of an artist (See figures 3.3) once again in the previous chapters. Drawings are made with markings that build up next to each other or on top of each other resulting in a finished artwork. During styles such as crosshatching, marks are made in one direction and then more are made on top of the previous markings in the opposite direction (See figures 4.1).

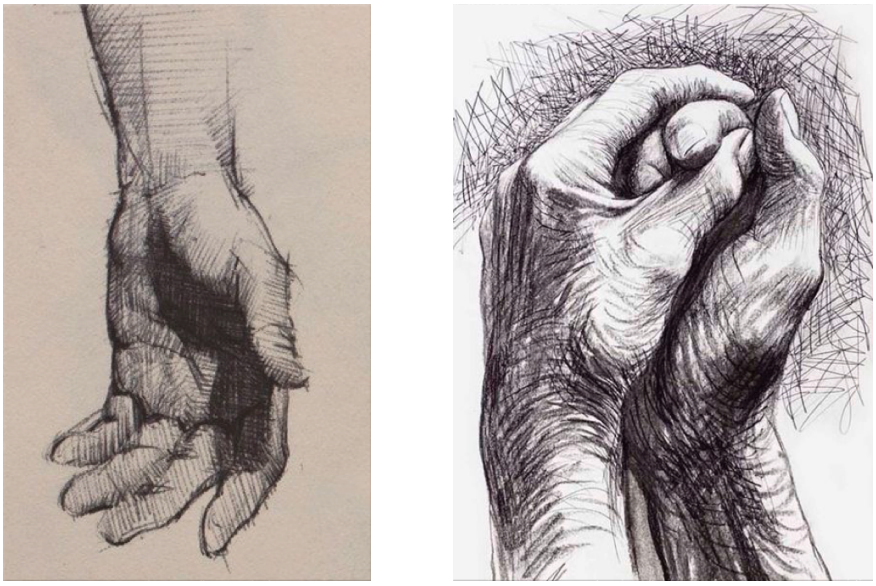
Figure 4.1: Woman Portrait by Igor Lukyanov

To create this effect using the method introduced in the previous chapter, there must be control images that already has a cross-hatching texture. If the texture needs more or less markings, a new texture must be created that has the desired texture, resulting in potentially large quantities of similar textures. To replicate the layering process of the human hand, the original shading method can be refined by replacing the opaque textures with transparent textures. In this chapter, I will introduce a method using transparent

control images to relieve the effort of creating multiple textures for each shade of darkness in the Barycentric shader.

4.1 Shader Development

While the Barycentric charcoal shader using opaque control images work quite well, the creation of the textures can be time-consuming. In order to create believable charcoal textures, the artist has to create each control image independently. Crosshatching or any other type of drawing style needs to have its own image and then have the image be manipulated for brightness correction. Such a process could take hours just to have a full library of control images for the shader to use. Furthermore, drawings can have multiple styles. Cross-hatching textures do not necessarily have to be straight lines, and they can run in all directions (See figures 4.2). If we consider all the different type of styles and lines, there needs to be a way to mix the control images so that different type of lines can be incorporated based on the illumination.



(a) Academia Study (b) "The Artist's Hands" by Henry Moore

Figure 4.2: Drawings of hands.

A further advantage of the transparent method is that, unlike shading with the opaque textures, shading with transparent textures can make use of a single control image instead of requiring multiple control images. By rotating or translating a single control image, multiple shapes and shades can be generated (See figures 4.3). Furthermore, the transparent control images can be layered over each other, which is very similar to the drawing process and how markings are made on paper.

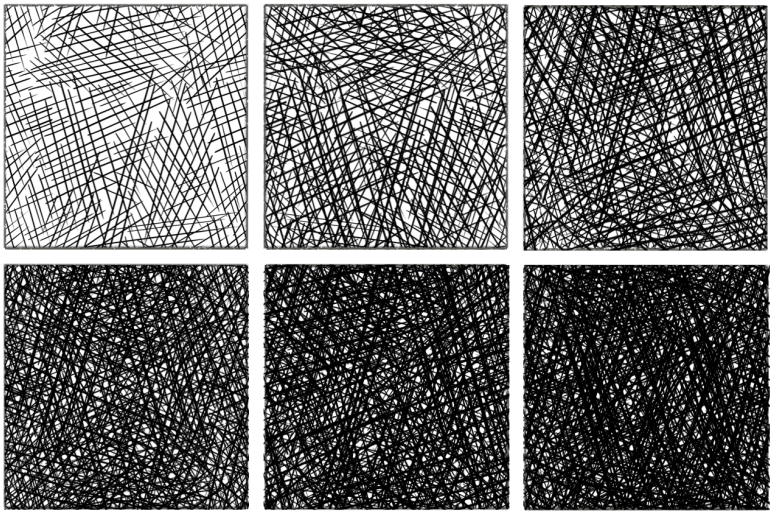


Figure 4.3: Transparent control images: Only one transparent texture is needed. It can be layered on top of itself with translation and rotation to create new and darker images

Development of this refined method is simple and is in a large part the same method as the Barycentric shader mentioned in the previous chapter with minor adjustments. Barycentric algebra utilizing the B-spline function $C(u,v) = \sum_{i=0}^{M} B_i(t)T_i(u,v)$ is is still used. However, the texture image $T(u,v)$ will no longer be a simple switch between one image to the next. Instead, it will layer control images or not layer them. When t is fully illuminated (t=1), a default paper texture will be used. Otherwise, a build-up of i transparent images will be used, on the point $C(u,v)$ depending on the illumination t.

Tri-planar projection is still the projection of choice for this method, as the only changes here are the way control images are handled.

4.2 Texture Creation

Only one texture is needed for the transparent texture method. However, this is only on a minimal level. If varying styles are desired, then more textures could be created (See figures 4.4). For this method, the textures are created using an image editing software Adobe Photoshop. This is just to ensure very clean markings, where any areas without markings (white paper areas) can be interpreted to be transparent areas.

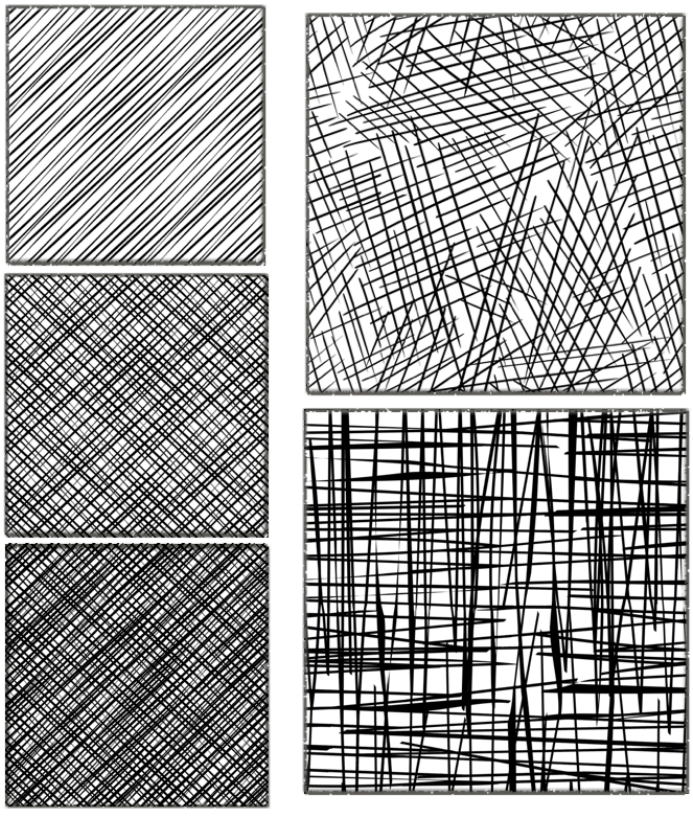


Figure 4.4: Transparent control images can be any style (right). When layered over each other and flipped, rotated, or translated, they can form darker patterns (left).

The markings and strokes have been placed in such a way that they have relative uniformity in density of lines. We do not want certain areas to be too dark or too bright confuse the viewer upon shader application, so uniformity and even distribution of lines are important (a topic that I will talk about this further in the next chapter).

Similar to the opaque texture creation, the transparent textures are also made repeatable on all edges so that no seams will be present on the rendered model. However, only one transparent texture needs to be used, so there is no need to create a control image for each value of brightness.

4.3 Shader Implementation

Like the previous opaque texture method, implementation using the transparent control images is done through the visual node network in Maya as well. However unlike the previous method, only one texture needs to be brought into the shader, which can be visually seen to be more straight-forward with less clutter (See figures 4.5).

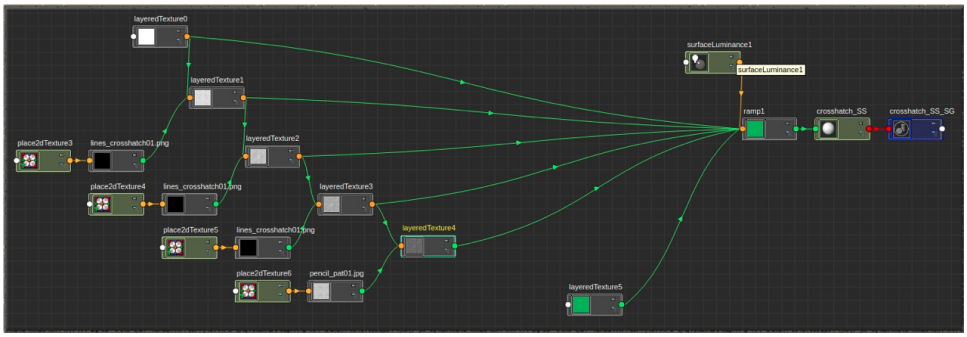


Figure 4.5: Creating Barycentric shader with transparent control images using ramp function from node network

The ramp node continues to be used for the Barycentric approach. However, a new node called "layeredtexture" is now used with the transparent texture shader so that I can

now layer my control image on top of itself within the ramp (See figures 4.5). Using this node, I am multiplying the transparent images with each other, which gives me a considerably darker texture with each new layer. The "place2DTexture" node is attached to each layer to allow a change in translation and rotation to occur so that the markings on the control image is not perfectly aligned with each other. This will give a crosshatching effect as the illumination of the model surface changes.

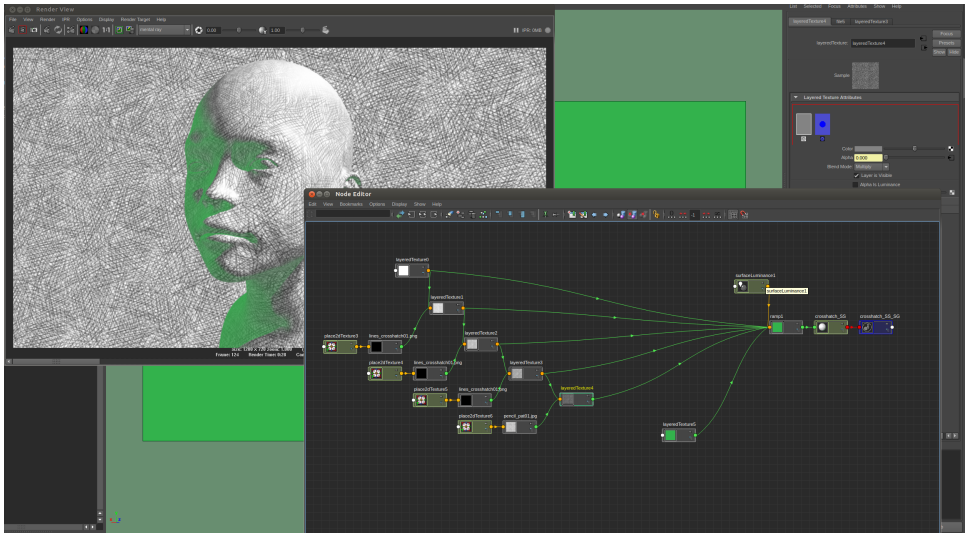


Figure 4.6: Test render in Maya of a boy model using transparent control images (left). Node network setup with a green constant set as a substitute for one of the control images (right).

The user can use this shader with multiple control images as well as a base image to create different styles of crosshatching. This method has great potential and can be used on a number of different drawing styles and mediums. Color can also be manipulated within the shader within each layer for any desired artistic direction (See figures 4.6).

4.4 Results

The Barycentric shader using transparent control images is even more promising than the previous method. Not only does the final render look believable, the process is much simpler. By having the user only needing one control image instead of an entire set, much time is saved in the process. Furthermore, the shader allows greater flexibility for the user, allowing multiple layers to be created within the shader itself. It is a much more stream-lined approach that follows the artist's hierarchy, which therefore gives a much more realistic image that mimics the drawing of an artist.

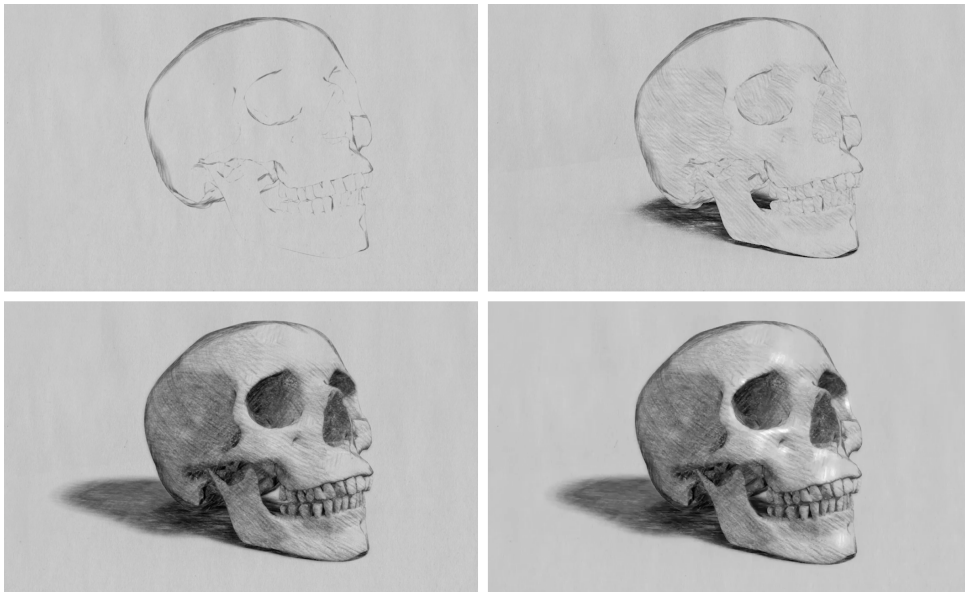


Figure 4.7: Render of a skull model using transparent control images. The artist can choose to render the model in any given parameter of the shader by layering transparent control image(s).

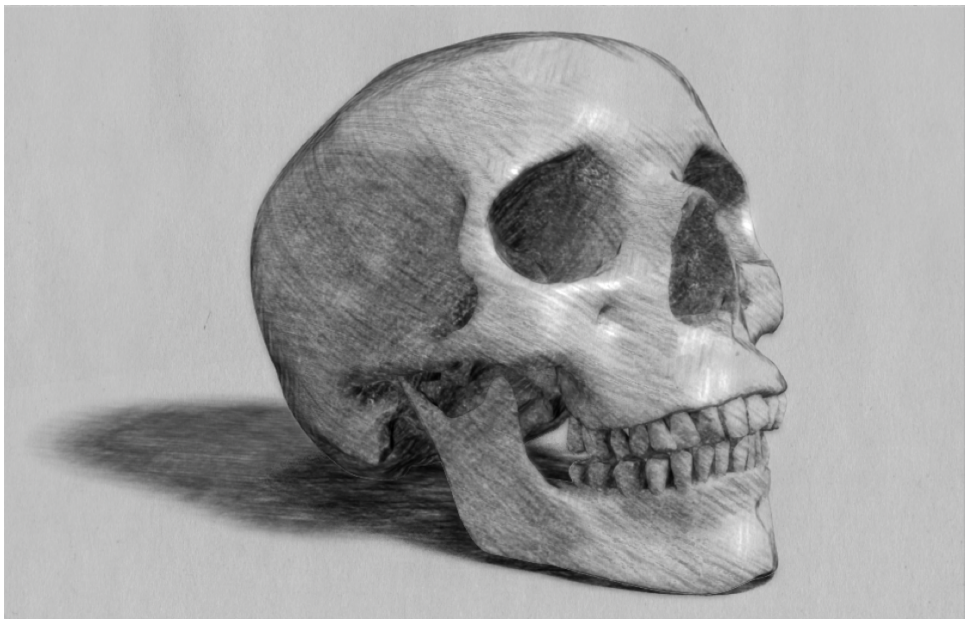


Figure 4.8: Final render of a skull using the transparent control texture method

5. TEXTURE SYNTHESIS

In addition to the Barycentric methods proposed in the previous chapters, I will also introduce a filter method that creates control images for the user in crosshatching styles by using an image synthesis algorithm. This algorithm will generate new images based off of an input image that is within the goal brightness chosen by the user without losing texture detail.

5.1 Deficiencies Without Texture Synthesis

The main problems when choosing control images for the charcoal Barycentric shader is that the image must meet certain conditions for it be be usable, which may take considerable time for the user to find or to create. The conditions are as follows:

1. Markings need to be evenly dispersed in density throughout the image, so that there are no striking bright or dark areas

2. Image needs to be within a user-chosen brightness value

3. Image should be a repeatable image that tessellates in all directions without seams

With such conditions in mind, not any image can be suitable to be a control image for the shader. The user may find themselves using considerable time just creating a control image that satisfies all conditions. Using just the naked eye, users may not be able to successfully create an image that meets the first condition, where the image has uniform strokes across the image.

Furthermore, with the current methods, users are confined to just existing styles. With the texture synthesis algorithm, non-existing styles can be created.

5.2 Methodology

In addition to the theoretical considerations discussed in previous sections, the practical issue of inconsistencies in texture brightness must be considered. To deal with this issue, a texture synthesis algorithm is implemented to construct on average constant color crosshatching textures based on a single crosshatching image (See Figure 5.1). While any image can be used, a tessellating or tiling image is recommended. The first step to this method is to perform a brightness averaging algorithm on the image. In the texture synthesis algorithm, the brightness values of a pixel of the original image changes so that the brighter areas become darker and the darker areas become brighter, as shown in Figure 5.1. After performing the texture synthesis algorithm, these modified textures can be mapped to objects using the tri-planar projection method.

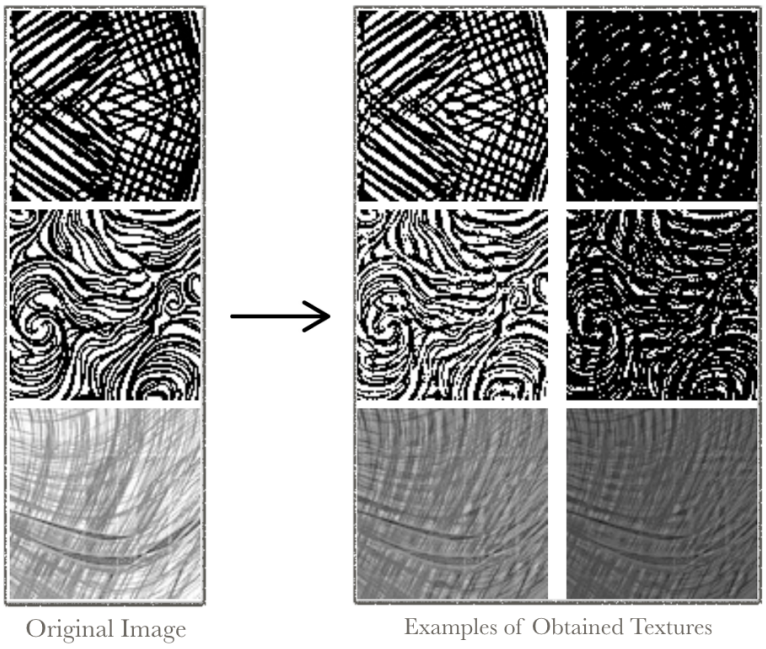


Figure 5.1: Control Image creation using the texture synthesis algorithm

The texture synthesis algorithm consist of two stages. In the first stage, a region is assigned around every pixel. The shape of the region and computational weights are controlled by a user-defined kernel. Based on the shape and weights of these regions, an average color in that region is computed around the pixel. In the second stage, a simple decision point is made based on a goal color assigned by the user: if the average color of the region is darker than the goal color, the color of the pixel is set to a lighter color. If the average color of the region is not darker than the goal color, the color of the pixel is set to a lighter color.

5.3 Implementation

To implement the texture synthesis algorithm, the scripting language "Processing" was used. In "Processing," a dilation or an erosion filter is applied to an input image. The user sets a kernel used to define a region size. Within the region, the program will choose dilation or erosion based on the brightness of the region.

The specific algorithm used for every pixel is shown in the equation below, where C is the pixel brightness, u and v are the coordinates of a specific pixel, $C_{region-avg}$ is the calculated average pixel brightness of the corresponding region to the pixel, $C_{goal-avg}$ is the goal average brightness, max(C) is the brightest value in the region, and min(C) is the dimmest value in the region:

$$C'(u,v) = \begin{cases} C(u,v) + C_{region-avg} - max(C) & \mathbf{C}_{region-avg} \geq C_{goal-avg} \\ C(u,v) + C_{region-avg} - min(C) & \mathbf{C}_{region-avg} < C_{goal-avg} \end{cases}$$

In a single iteration of the texture synthesis algorithm, the above algorithm is repeated for every pixel of the texture. The user can repeat this process over any desired number of iterations. However, on a practical level, a sufficiently high number of iterations will eventually lead to a completely gray and aesthetically-bland texture. It is therefore completely

up to the user to choose the number of iterations, the goal brightness, and the kernel size for a desired texture (See Figure 5.1).

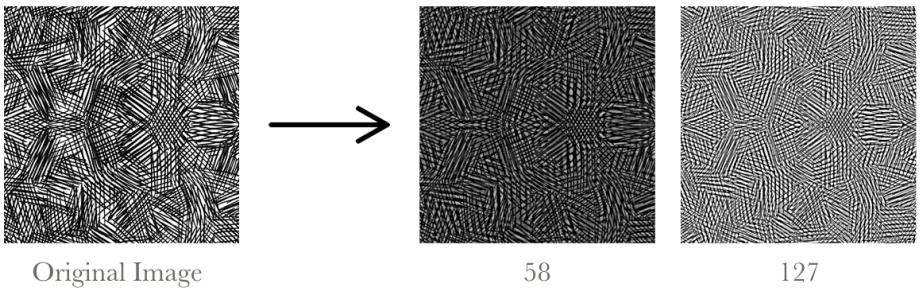


Figure 5.2: Original Image (left) and rendered image from the algorithm based on the goal values chosen by the user: 58 and 127 (right)

5.4 Results

The texture synthesis algorithm produces promising hand-drawn effects from any given image. It provides not only tremendous artistic control and save time for the user, but it also produces artistic styles that have never been done before. The utilization of texture synthesis with Barycentric shader proves to be a powerful unison.

The following are the rendered results using the texture synthesis approach and then applied to the Barycentric shader.

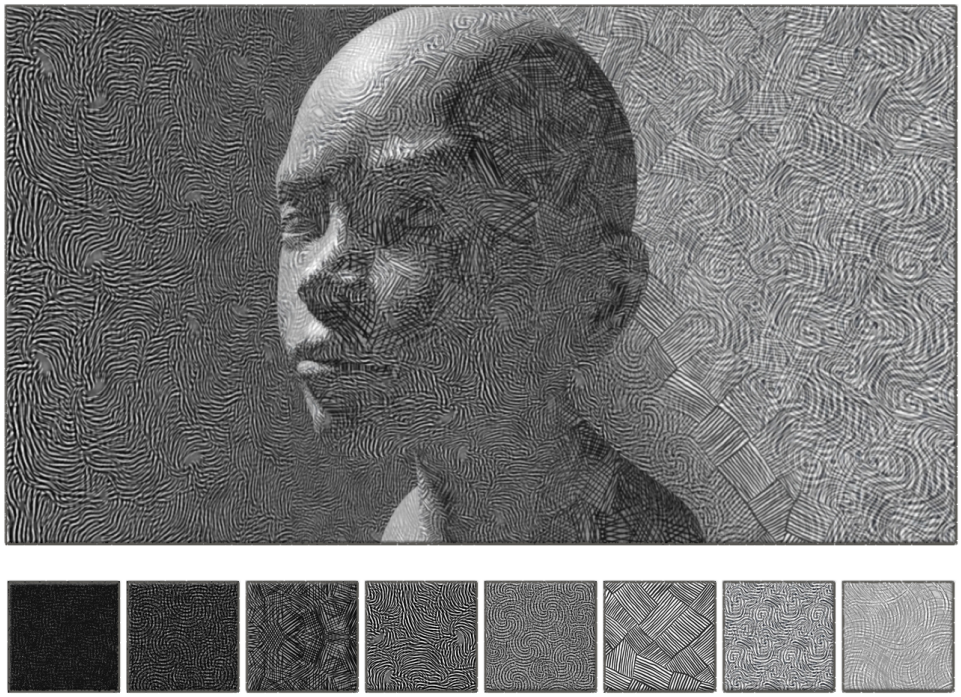


Figure 5.3: Boy Render using textures created by the image synthesis algorithm (top) and the textures used from the algorithm (bottom)

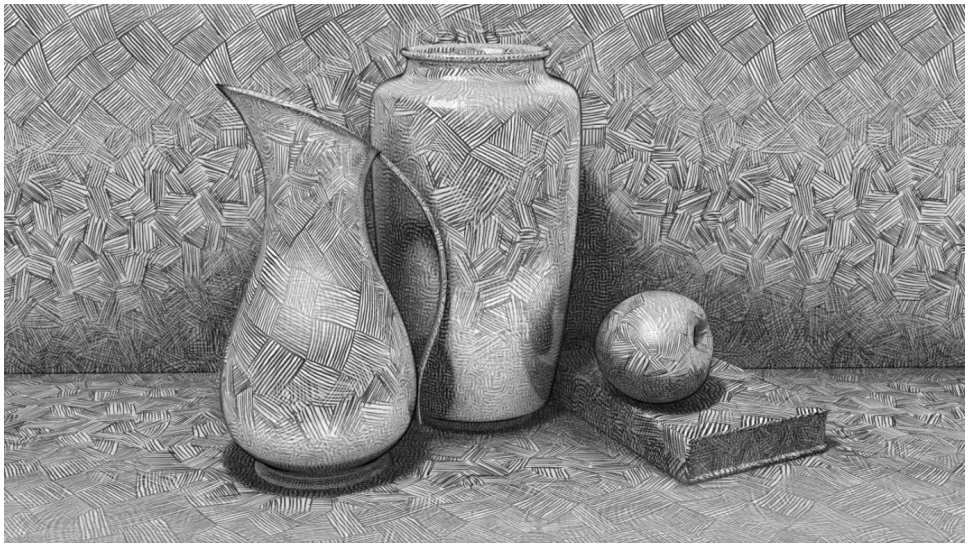


Figure 5.4: Vase Render using textures created by the image synthesis algorithm

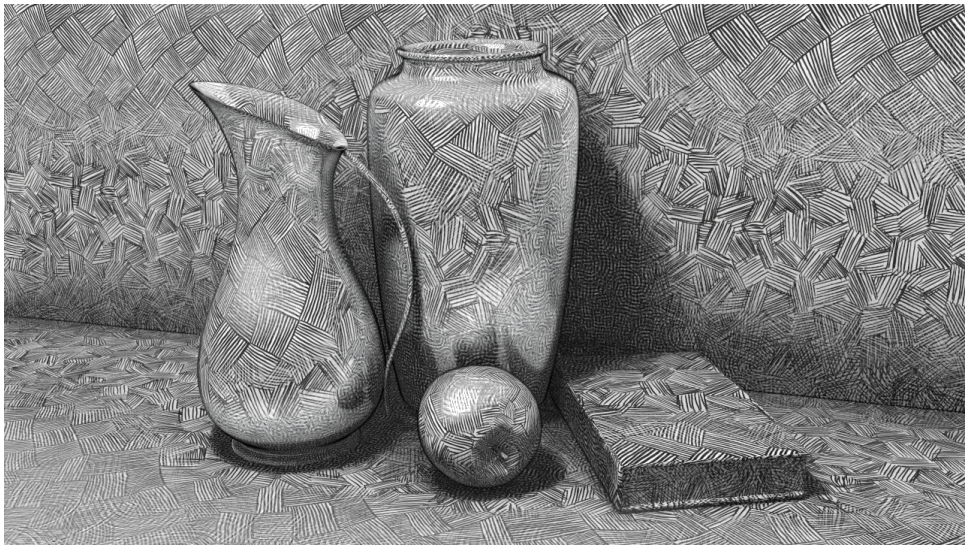


Figure 5.5: Vase Render using textures created by the image synthesis algorithm

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

The Barycentric shader method allows for easy implementation. It is the closest method to allowing artist to "draw" their shader, giving artists the freedom to produce any artistic style on a 3D geometry. This methodology also makes the shader highly controllable as the artist has complete authority over diffuse, contours, shadow, reflection, and specular areas. Additionally, the method is versatile as it can be used for cross hatching, smudging, and even painting depending on the desired style given by the control images.

6.2 Further Study

While the current stage of the research gives very promising results, more research needs to be conducted on how these methods apply to color. The Barycentric shading with transparent multi-textures approach currently shows that the layering of transparent textures on top of each other produces a nice mixture of colors, but they may not be entirely accurate colors that the artists wants. It is a trial and error process to produce the perfect end-result (See Figure 6.2).

The image synthesis algorithm can also be improved so that other artistic styles can be incorporated. Currently, it uses dilation and erosion to mimic the artist hand of cross-hatching. It would be extremely interesting to see if styles such as painting, water-color, and other mediums can also be incorporated.

Furthermore, more research is needed to improve the image synthesis program, where the textures produced can look more rough and textured. Hopefully, the algorithm can be improved so that the result may look as natural as the product of an artistic hand-drawn texture.

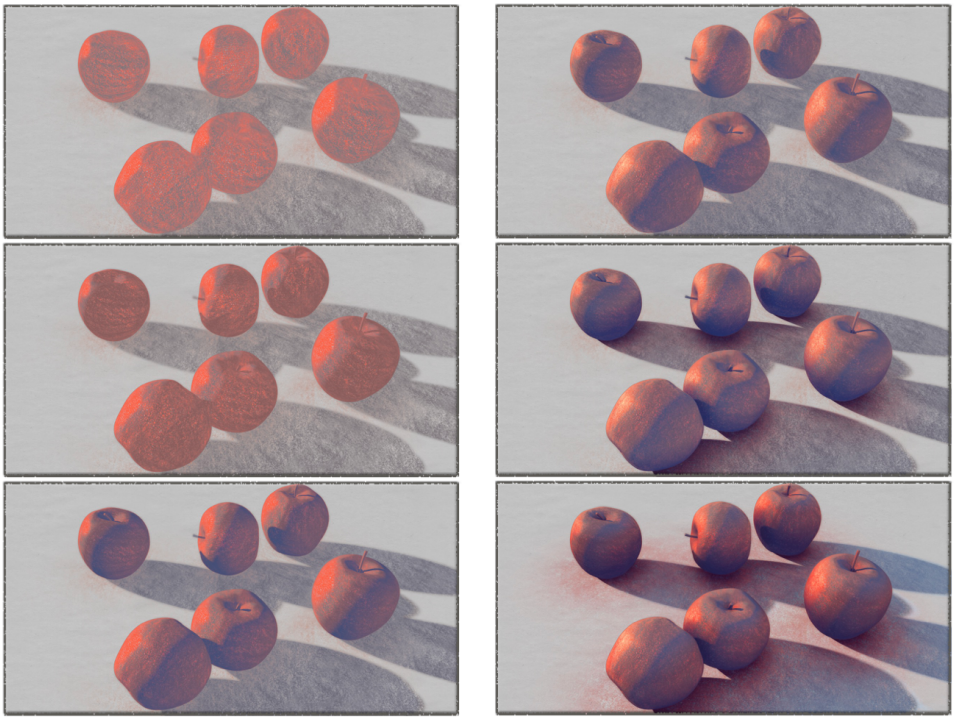


Figure 6.1: Barycentric Shader following the artist's hierarchy while using colored control images

Figure 6.2: Details of textures that are obtained using the image synthesis algorithm that produced lighter or darker textures that resemble original image. Algorithm works not only for black and white images, but also gray-scale and color images.

REFERENCES

[1] R. L. Cook, "Shade trees," *ACM Siggraph Computer Graphics*, vol. 18, no. 3, pp. 223–231, 1984.

[2] P. Hanrahan and J. Lawson, "A language for shading and lighting calculations," in *ACM SIGGRAPH Computer Graphics*, vol. 24, pp. 289–298, ACM, 1990.

[3] W. Kemp, "How to shade a drawing (light and shadow : Part 2)," Jan 2016.

[4] F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz, "Where do people draw lines?," *Communications of the ACM*, vol. 55, no. 1, pp. 107–115, 2012.

[5] E. Akleman, S. Liu, and D. House, "Barycentric shaders: Art directed shading using control images," 2016.

[6] S. Liu and E. Akleman, "Chinese ink and brush painting with reflections," in *SIGGRAPH 2015: Studio*, p. 8, ACM, 2015.

[7] A. Majumder and M. Gopi, "Hardware accelerated real time charcoal rendering," in *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pp. 59–66, ACM, 2002.

[8] M. C. Sousa and J. W. Buchanan, "Computer-generated graphite pencil rendering of 3d polygonal models," in *Computer Graphics Forum*, vol. 18, pp. 195–208, Wiley Online Library, 1999.

[9] Y. Du and E. Akleman, "Charcoal rendering and shading with reflections," in *ACM SIGGRAPH 2016 Posters*, p. 32, ACM, 2016.

[10] S. Grabli, E. Turquin, F. Durand, and F. X. Sillion, "Programmable rendering of line drawing from 3d scenes," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 2, p. 18, 2010.

[11] C.-R. Yen, M.-T. Chi, T.-Y. Lee, and W.-C. Lin, "Stylized rendering using samples of a painted image," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 468–480, 2008.

[12] C. Tietjen, T. Isenberg, and B. Preim, "Combining silhouettes, surface, and volume rendering for surgery education and planning.," in *EuroVis*, pp. 303–310, 2005.

[13] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.