5-2017

Procedural Spanish Moss Renderman Shader

Summer O'Neal Benton
Clemson University, sobento@clemson.edu

Procedural Spanish Moss Renderman Shader

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Fine Arts
Digital Production Arts

---

by
Summer O'neal Benton
May 2017

Accepted by:
Dr. Jerry Tessendorf, Committee Chair
Dr. Donald House
Professor David Donar

Abstract

This thesis discusses the creation of computer generated photorealistic Spanish moss, a plant that grows in trees in the American Southeast. In order to replicate the plant, a Renderman procedural shader was used that included hair-like qualities in order to replicate the surface properties of Spanish moss. The necessary attributes, how they are used in the code, and final results of the shader applied to a moss model are included in this paper.

Table of Contents

List of Figures

Chapter 1

Introduction

There are many types of flora that inhabit only particular places in the world. An example of one of these plants is Spanish moss, a plant that is native to the humid areas of the Southeastern United States, but can grow in other tropical like areas. Because of its limited indigenous locations, the moss does mark the landscapes uniquely. The long grey curly hair-like strands of Spanish moss hanging from low lying branches of live oaks adds to the hallmark image of the gothic South. For an artist, one of the many challenges of attempting to bring a unique piece like the Spanish moss into the computer graphics world is to capture the different properties that make the subject unique. The knowledge and understanding of the different aspects that contribute to a piece in the natural world are necessary in order to be translated into code for the computer graphics world. The focus of this thesis is to bring Spanish moss into a computer graphics environment using a procedural shader that will help artists create realistic and accurate surface properties for this plant. The artist can use the shader with its default attributes or change the attributes of the shader in order to meet their artistic vision.

Before attempting to replicate an organic piece in CG, it is important to know and understand more about the natural properties of the object. Spanish moss is grey, curly, tangled, and hair like, and grows in clumps on the branches of trees (see Figure 1.1). It commonly grows in the Live Oak and Cyprus trees of the wetlands although it can grow in slightly dryer locations. It is most easily identified by its grey mass entangled in the leaves and on the branches of trees.

Since Spanish moss grows intertwined in the branches of trees, one may think that Spanish moss is connected to the tree and perhaps even living off of the tree's nutrients. However, Spanish

Figure 1.1: Live oak with Spanish Moss [1]

moss is not a parasitic plant. Rather it is an epiphytic plant and only lives resting on the tree's branches. Spanish moss has no roots, but rather gets the nutrients for its survival from rain as it falls from the sky and dust flowing through the air.

Although the name "Spanish moss" would lead one to guess its native homeland and its scientific classification, it is not from Spain or a type of moss. Rather, it is a flowering plant in the Bromeliaceae (pineapple) family. Unlike real moss which reproduces using spores, Spanish moss is a flowering, seed producing plant.

The South is known for its various folk tales and legends and tall tales. This applies to Spanish moss as well and there are several, similar stories of how the plant originally gets its name. Although there are many different legends, most interpretations seem to agree that the grey strands hanging from the trees in clumps reminded various peoples of the beards of early Spanish settlers and explorers. Although the original reason for the name has been lost to time, the name "Spanish moss" has remained.

Knowing and understanding Spanish moss and how it reacts to its environment is essential in order to replicate these plants in a graphics worldspace. The surface attributes of the plant are more than its grey color, but rather how it reacts to light and the other objects in its environment

(Figure 1.2). Since Spanish moss is hairlike in its structure, it is essential to understand how a strand of hair reacts to and interacts with light as well. Understanding these pieces will add to the realism of the Spanish moss model after the procedural shader has been applied.

The shader needs to incorporate the surface properties of the Spanish moss, but it also needs to allow for control and freedom over these attributes so that the artist can use the shader to acheive a certain visual design. Although the Spanish moss is usually grey like in color most of the time, Spanish moss does change colors when it rains and in other environmental conditions. Also, the project that the artist may be working on may require a Spanish moss that is stylized in a way where it is has more green surface color showing or a different subsurface color or some other change that may not occur in nature in order to follow the artistic vision of the project. The shader needs to be designed in such a way that the artist can change these attributes.

This paper discusses these challenges and how they were confronted. Chapter 2 discusses the components of the shader. This chapter talks about the shading language used and the methods used for the shader to achieve the necessary surface properties. Chapter 3 explains how the shader was implemented and how the methods discussed in the previous chapter were incorporated into the code. Chapter 4 covers the final results of the shader and how the surface changes when certain attributes are altered. Finally, Chapter 5 discusses how this shader can be used in future work.

Figure 1.2: Spanish Moss interacting with sunlight [2]

Chapter 2

Components Necessary to Recreate Spanish Moss

In order to successfully translate the Spanish moss into a language the computer can understand, a a shading language must be chosen. The procedural shader for Spanish moss is written in the Renderman Shading Language and is rendered in Pixar's photorealistic renderer, Renderman. In order to incorporate the natural surface properties of the Spanish moss, particular attributes were replicated and incorporated into the shader code. Hair shader attributes (Bidirectional Reflectance Distribution Function, Bidirectional Transmittance Distribution Function, and subsurface scattering), Fractal Brownian Motion noise, and the portions of the Pixar Coral Reef shader are a few of the attributes that are included in the Spanish moss procedural shader. Using these features will help produce code to replicate the physical portions of Spanish moss.

2.1 Renderman Shading Language

In order to create a photorealistic render of Spanish moss, a photorealistic way to render a three dimensional image is necessary. "Renderman is first of all a scene description methodology: a comprehensive way to describe objects, scenes, lights and cameras so that a computer can create images from them." [3]. Shaders for Renderman are written in the Renderman Shading Language which is a C-like coding language. It was originally introduced in 1990 and incorporates the idea

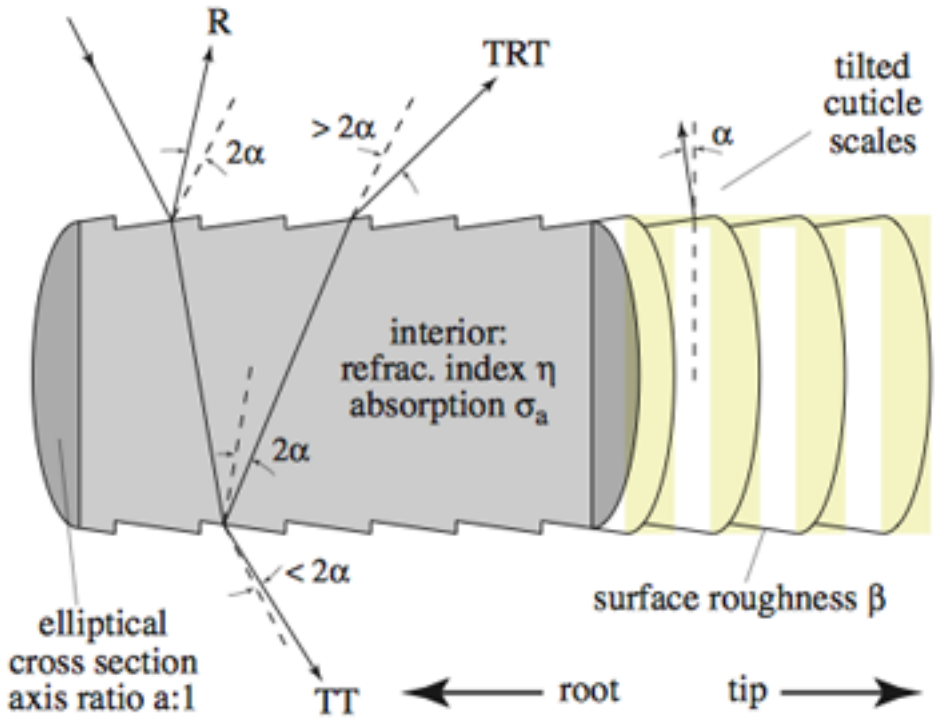


Figure 2.1: Dissection of a strand of hair [5]

that small pieces of code can be "plugged into other pieces of code" which is called a shading tree [4]. Renderman is a robust renderer that has been used for numerous feature and short films since its original development. It allows for artistic control and manipulation of attributes in the shaders in order to generate the best surface properties for a model. Although it has been used in the industry, Renderman has been recently released for free for public use. Renderman Shading language is designed to be relatively easy to learn and use for people who are proficient in coding or have been newly introduced to programming. Using the Renderman Shading Language allows for an artist with limited or extensive programming language experience to understand the concepts and principles of the language, and to be able to create photorealistic images.

2.2 Spanish Moss and Hair

When observing Spanish moss, one can observe how the grey, silvery wisps appear like thicker strands of curly hair. Since Spanish moss and hair have a similar appearance, one can hypothesize that they both have similar surface properties. Understanding how a light ray interacts with a strand of human or animal hair can help one understand how a light ray can interact with a strand of Spanish moss since they are both very similar structures. When a light ray shines onto a strand of hair, there are three paths that the light can travel. The light ray can hit the strand and bounce off of the strand immediately. The light ray can hit the strand, enter the strand, and exit the strand. Or, the light ray can enter the strand of hair, bounce multiple times within the strand of hair, and then exit the strand (Figure 2.1). These three paths are necessary to compute mathematically within the shader in order for the Spanish moss to react to light properly. The three ways to translate these paths into shader code language are Bidirectional Reflection Distribution Formula, Bidirectional Transmittance Distribution Function, and Subsurface Scattering.

2.2.1 BRDF

The first of the possible paths for a light to travel when it encounters an object is to reflect off back into space as soon as it touches the surface of the object (Figure 2.2). The more opaque an object is, the more light is reflected off of an object. The way in which to calculate how much light is reflected off of a object's surface is called BRDF. The directions of these rays of light as they shoot back into space affect the specular highlight surface property of the object. The different distributions of light as they reflect off result in a more broad or tight specular highlight. This specular highlight and light reflectance can give more information about the size and surface texture of the model.

2.2.2 BTDF

Another possible path for a light ray to take is called transmittance. Transmittance occurs when a light ray hits an object, keeps moving through the object, and exits out of the object, shooting back into space. The rays coming away from the object can also contribute to the specular surface properties of the object (Figure 2.3). The calculation of how much light is transmitted in a surface is called the BTDF. Transmittance is very similar in concept to reflectance except that in

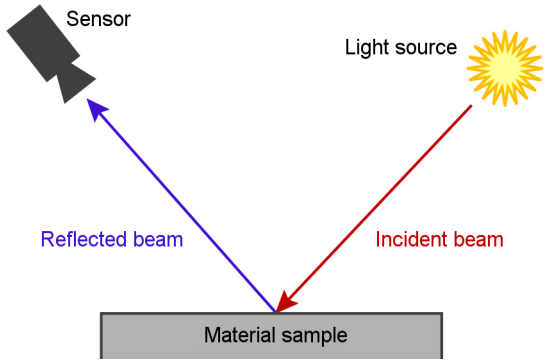


Figure 2.2: Light reflecting off of a surface. The measurement of this reflected light is BRDF [6]

transmittance, light rays exit after entering the object rather than reflecting as soon as it touches the surface like the way in which reflectance light rays travel.

2.2.3 Subsurface Scattering

Additionally, the light ray can take the path of entering the surface, bounce around within the object, and then exit the surface returning back into space. This ricocheting of light rays beneath a surface of a material is called Subsurface Scattering (Figure 2.4). An example of this principle is when a person shines a flashlight towards their hand. The light rays shine and penetrate the skin of the hand. The rays then bounce around within the muscles and blood vessels of the hand. After bouncing nd reflecting beneath the surface of the hands, the light rays then exit out of the skin and shoot back into space. Since the muscles and blood of the hand are red, the hand has a reddish light or glow shining around the edges of the fingers and hand (Figure 2.5). The thinner the portions of the hand are (around the fingers and outer edge of hand), the more visible the light glow is. Subsurface scattering occurs in hair and fur as well as many other organic objects. Using Subsurface Scattering allows for a model to appear more photorealistic and lifelike and less like a geometric model for computer graphics.

2.3 Fractal Brownian Motion Noise

Fractal Brownian Motion Noise, abbreviated as “FBM noise,” is a method of creating noise by using octaves in a loop to differentiate the frequency and amplitude each time it goes through

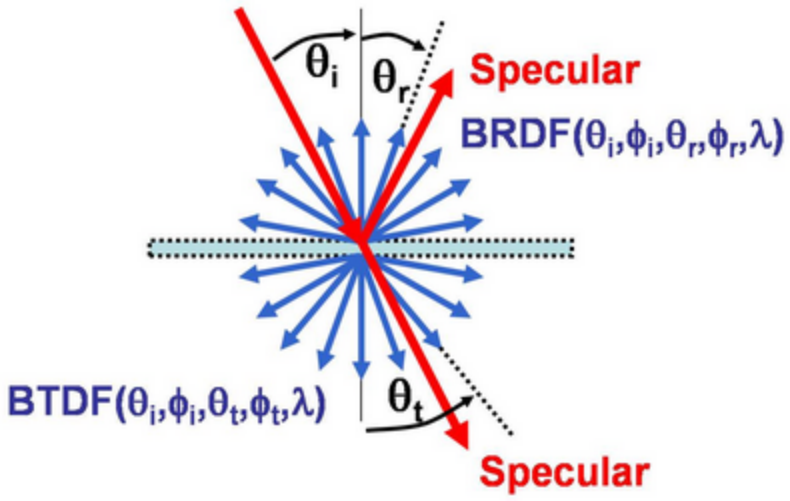


Figure 2.3: Light reflecting and transmitting through a surface. The measurement of this transmitted light is BTDF [7]

the loop. Octaves tell the code how many times to go through the loop and change the frequency and amplitude resulting in a differentiation and "randomness." This can create a random like noise that is controllable by the artist and appears the same every time the code is compiled rather than a random() function. A random function creates randomness by using time as a seed. A random() function is uncontrollable and changes the appearance of the noise each time the shader is compiled. Using FBM noise provides the desired colored differentiation in the shader by using the noise it produces.

2.4 Pixar Coral Shader

Pixar developed a versatile procedural Renderman shader for coral for their film, "Finding Nemo." This shader allows for the artist to create different variations and types of coral by only changing the attributes of the shader rather than having to create different shaders for different coral types. In order to include these attributes in the developing shader, understanding of some necessary anatomy of a coral is required. For example, coral polyps are an animal related to jellyfish and anemones (Figure 2.8). The polyps themselves are clear, but they build up a protective barrier made of limestone. The actual color of coral and coral reefs comes from the types of algae that lives in the reefs. In order to include these attributes in the shader, the coral shader included the idea of

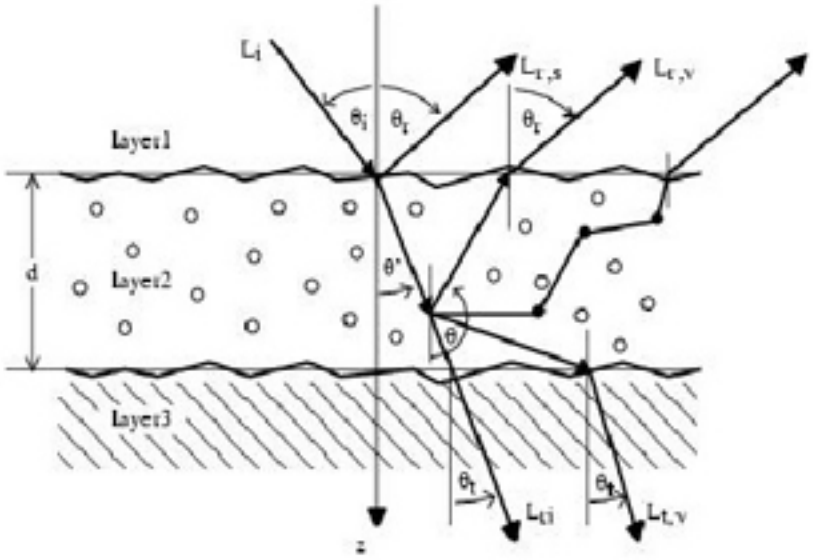


Figure 2.4: Subsurface Scattering Diagram [8]

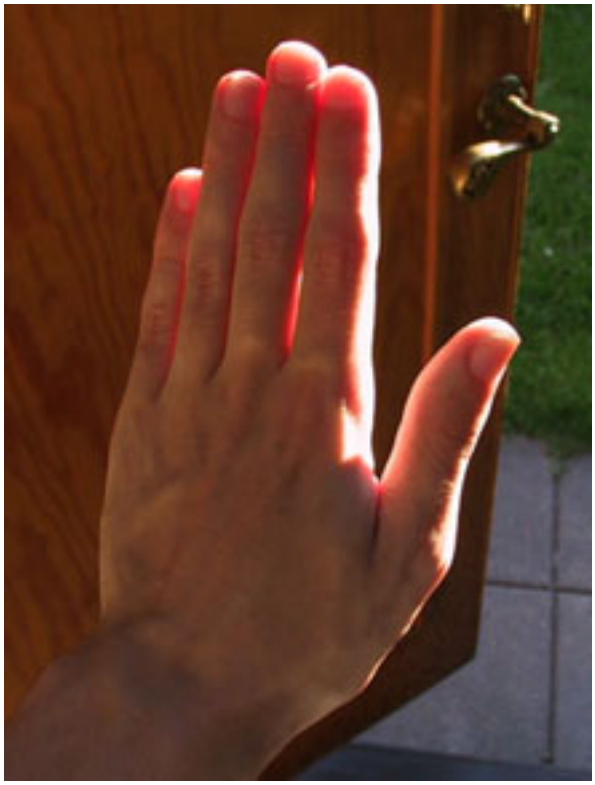


Figure 2.5: Subsurface Scattering in hand [9]

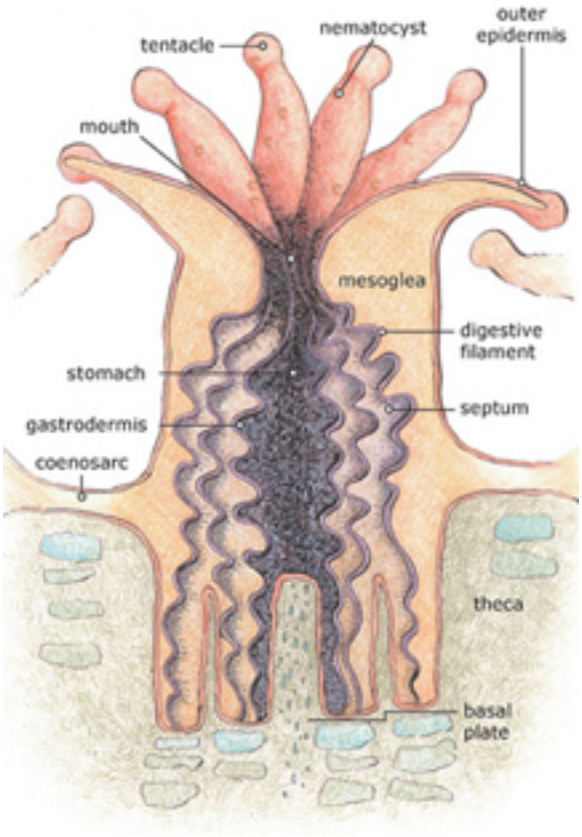


Figure 2.6: Anatomy of a Coral Polyp [11]

Figure 2.7: Layers of Coral Reef for Pixar's coral reef shader [10]

breaking the shader into three portions, the bone, velvet, and polyps. Since the coral polyp organism has no color, the artist is actually coloring the limestone protective barrier that is being influenced by the algae's color. In the shader, the artists call this layer the "bone" (figure 2.7). The second layer is called the "velvet" layer and brings the "soft-looking main surface of the coral" [10]. Lastly, the last layer to the shader is to replicate the bumpy, bulbous extrusions on the surface of the coral. Since coral polyps are related to jellyfish, they have a great amount of subsurface scattering in the surfaces (Figure 2.8). This velvet like layer in coral reef is very similar to a layer of soft fuzziness of Spanish moss. Since the coral reef shader is extremely versitile and was designed to surface multiple types of coral, part of this shader can be assimilated to the Spanish moss shader.

2.5 Displacement

In nature, Spanish moss is not smooth, but instead has a scale-like texture to it. In order to create this bumpiness of Spanish moss, one can create changes in the model. However, this will create more faces on top of multiple strands of hair. If the number of subdivisions a model has is too high, Maya will run much more slowly and crash. It also will more tedious for the artist to work in Maya because of how slowly the program is running. In order to create a layer of raised texture, a displacement map will be used for the Spanish moss shader. A displacement map uses an image or noise or some other procedural method to create the raised and indented areas. Displacement uses the map in order to change alter the surface points of the model to match the displacement map. This displacement is revealed at render time and does not slow down Maya by adding multiple subdivisions to the model itself. For the displacement in the Spanish moss shader, a form of turbulence will be used. Turbulence is another method of creating random like noise by using changing the amplitude and frequency using a for loop. In conclusion, the attributes mentioned in this chapter will be included in the Spanish moss shader and the implementation in code will be described in the next chapter.

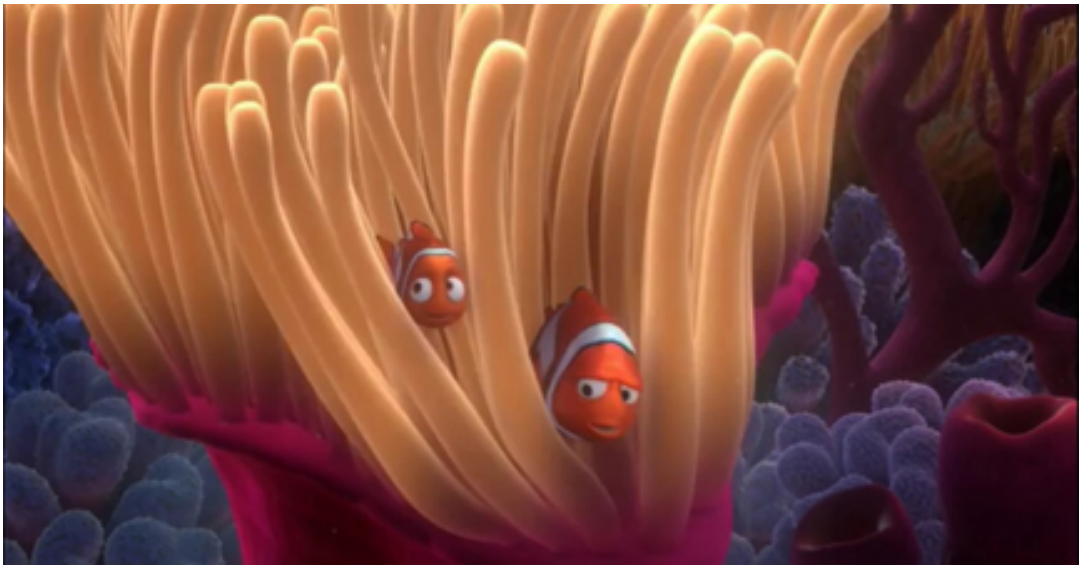


Figure 2.8: Sea Anemone in "Finding Nemo" Pixar 2003 [12]

Chapter 3

Spanish Moss in Shader Form

This shader is set up as a Renderman coshader so that the surface and displacement parts are compiled into one shader. The rendered file (.slo file) is then inserted into the shader input section of the Renderman surface in Maya. Without using a coshader, the displacement and surface shaders must be separated into two separate files and then be uploaded to separate file nodes and fed separately into the one renderman surface shader. Using a coshader will result in a more manageable artist workflow.

The coshader for Spanish moss has a class with multiple functions within it. The Spanish moss shader is set up like the following

- class spanishMoss()
  - private float brdfOrenNayer()
  - private color subsurfaceScattering ()
  - private private float btdfOrenNayer()
  - private color velvetFactor ()
  - private float fbmNoise ()
  - public void displacement ()
  - public void surface ()

The private functions are functions that the artist creates in order to achieve a particular task within the shader. The public functions are functions that are for the surface and displacement

functions that use the private functions the artist created in order to create surface color, model displacement, etc.

The first portion of the shader is the function that creates the diffuse and reflection aspects of the shader. This portion of the code provides the shader with the Bidirectional Reflectance Distribution Function needed in order to provide the moss with one of the proper interactions of light rays. We will also need BTDF and subsurface scattering later in the code. The mathematical code for the Oren-Nayer model is taken from an algorithm described in "Physically Based Rendering: From Theory from Implementation" (Figure 3.1). Translating this mathematical function into code the computer understands involves defining A, B, alpha, beta, theta i, and theta o. The following code defines these variables as floating point variables and computes the formula.

$$f_r(\omega_i, \omega_o) = \frac{\rho}{\pi} \left\{ A + B \max(0, \cos(\phi_1 - \phi_0)) \sin\alpha \sin\beta \right\}, \tag{3.1}$$

where if σ is in radians,

$$A = 1 - \frac{\sigma^2}{2(\sigma^2 + 0.33)} \tag{3.2}$$

$$B = \frac{0.45\sigma^2}{\sigma^2 + 0.09} \tag{3.3}$$

$$\alpha = \max(\theta_1, \theta_0) \tag{3.4}$$

$$\beta = \min(\theta_1, \theta_0) \tag{3.5}$$

Translating this mathematical function into code the computer understands involves defining A, B, alpha, beta, theta i, and theta o. The following code defines these variables as floating point variables and computes the formula.

```
private float brdfOrenNayer (vector Ln, V, Nn; float sigma)
{
    //formula is "1/PI (A + B max(0, cos(thetai -thetao) sin(a)*tan(b)"
    float A = 1.0 - (sigma * sigma/(2.0 * (sigma * sigma + 0.33))); //defines variable A
    float B = 0.45 + sigma * sigma /(sigma * sigma + 0.9); //defines variable B
    float cosi = abs(Nn.V);
```

```
    float thetai = acos(cosi); //defines variable theta i
    float coso = abs(Nn.Ln);
    float thetao = acos(coso); //defines variable theta o
    float a = max(thetai, thetao); //defines variable aplpha
    float b = min (thetai, thetao); //defines variable beta
    vector Vperp = normalize(V - Nn * (Nn.V));
    vector Lnperp = normalize(Ln - Nn * (Nn.Ln));
    float cosphi = Vperp.Lnperp;
    float brdf = (1.0/PI) * (A + B * max(0.0, cosphi) * sin(a) * tan(b)); //completes formula
computations and gives the value to the variable brdf
    return brdf;
}
```

The formula is computed and its result is given to the variable brdf. This brdfOrenNayer function is called later in the surface portion of the shader.

The second function in this code is subsurfaceScattering() which computes a second way light rays interact with a surface, subsurface scattering. Although it is only five lines of code, it computes the necessary subsurface scattering in a more time efficient way. This code was adapted from the coral reef shader which was created for Pixar's "Finding Nemo" [10].

Next, the Bidirectional Transmittance Distribution Function is calculated in the shader. The BTDF can be computed using code similar to the pre-existing BRDF computations. In fact, this adapter [13] uses the brdfOrenNayer() that was created earlier in the shader. It computes the ray for the BTDF in the opposite hemisphere of the ray for the BRDF and its hemisphere. The code below translates the BRDF to BTDF adapter from "Phyisically Based rendering: From Theory to Implementation" into Renderman Shading Language.

```
private float btdfOrenNayer(vector Ln, V, Nn; float sigma)
{
    vector LnR = Ln - 2 * Nn * ( Nn.Ln);
    float brdf = brdfOrenNayer(LnR, V, Nn, sigma);
    float btdf = 1 - brdf;
    return btdf;
```

```
}
```

Next, the shader computes the mathematical calculations for a formula that creates the velvet layer of the Spanish moss. This velvet code was originally created for the same Pixar coral reef shader as was the code for subsurface scattering that was created earlier in the shader. The base of the Spanish moss is a certain color, but the velvet() adds a layer of soft velvety color on top of that base color. This provides the multidimensional of the moss. The following code provides these computations.

```
private color velvetFactor(vector Nn, V, Nfv, Ln, Nf)
{
    vector H;
    float sssEdgeDiffusion = 1.50;
    float sssEdginess = 10.00;
    color cSheen = (0.60, 0.60, 0.60);
    float sssEdgeMix = 0.50;
    color minC = (0.0, 0.0, 0.0);
    varying normal nbd = 0;
    vector mixN = (0.5 * Nn) + (0.5 * normalize(nbd));
    float cosine, sine;
    float normalDot = clamp(V.mixN, 0, 1);
    float nDot = min(-normalDot * sssEdgeDiffusion, 1);
    nDot = clamp(-nDot, 0, 1);
    color outerEdge = mix(cSheen, minC, nDot);
    vector lightView;
    float backscatter = 0;
    float roughness = 0.01;
    color velvet = 0;
    vector lv = 0;
    illuminance ( P, Nfv, 1.57079632679489661923 /* Hemisphere */ ) {
/* Retroreflective lobe */
    cosine = max ( Ln.V, 0 );
```

```
    velvet += pow ( cosine, 1.0/roughness ) * backscatter * Cl * cSheen;
/* Horizon scattering */
    cosine = max ( Nfv.V, 0 );
    sine = sqrt (1.0- sqrt(cosine));
    velvet += pow ( sine, sssEdginess ) * Ln.Nf * Cl * cSheen;
    }
return velvet;
}
```

The final private function in the shader is the fbmNoise() which computes the Fractal Brownian Motion noise. This function creates a noise that changes the noise as it goes through a loop for ten octaves. This noise is used to add differentiation in the color of the base of the model. No object in the natural world is one perfect, solid color. There is always some change and difference. This function is used to add some of that natural coloring to the shader surface color. The code for this fbm noise is the following:

```
private float fbmNoise (vector x; float fjump; float frequency; float roughness)
{
    float newfjump = 1;
    float newRough = 1;
    float value = 0;
    float i = 0;
    while(i < 10)
        {
        value += newRough * noise(x * newfjump * frequency);
        newfjump *= fjump;
        newRough *= fbmRoughness;
        i = i + 1;
        }
    return value;
}
```

The final portion of the shader uses these defined functions to create the displacement and

surface of the shader. The displacement uses a turbulence noise to create a bumpiness texture for the Spanish moss. This turbulence contributes to the scale like bumpiness to the model. The shader uses the following code to change the size and frequency of the displacement noise pattern as it goes through the for loop five times. The shader then changes the surface point (P) to meet the pattern generated by the for loop in order to calculate a new surface point (P). This changes the shape of the model which will result in a bumpiness pattern on the geometry.

```
float Km = 0.1;
float freq = 6.0;
float layers = 5.0;
string space = "object";
float hump = 0.0;
normal Nn = normalize (N);
point p = transform(space, P);
float j, f = freq, amplitude = 1.0;
for (j =0; j < layers; j += 1)
    {
    hump += (noise(p * f) - 0.5) * amplitude;
    f *= 2;
    amplitude *= 0.5;
    }
P = P - Nn * hump * Km;
N = calculatenormal (P);
```

The surface function finishes the shader. The surface computes the specular attributes, diffuse attributes, fbmNoise(), velvet(), and SubsurfaceScattering() to add to the final surface color of the Spanish moss.

Chapter 4

Spanish Moss Results

The code for the Spanish moss procedural shader results in a shader that allows an artist to control multiple attributes such as frequency, coloring, subsurface scattering strength, and more. Adjustments may be made, but the shader works as moss with the attributes set as is.

The current settings for the moss shader are a displacement depth of 5.0, Ksss of 1.0 and the subsurface scattering color is a light green, sigma of 0.01, specular roughness of .05, fjump of 2.2, frequency of 8.0, fbmRoughness of 0.510, and dispacement frequency of 5.0. I believe this is the best representation of Spanish moss because it captures the surface properties of Spanish moss well. The displacement depth of 5.0 gives the individual strands the bumpiness necessary to replicate moss scales. The subsurface scattering color and Ksss of 1.0 work to give the subsurface scattering without overpowering the diffuse colors of the moss. A light golden green and a light blueish green were chosen as the two diffuse colors to mix together with a fbm roughness of 0.05 to best replicate the coloring of Spanish moss. With the specular roughness and gamma calculated into the scene, the result is a representation of Spanish moss. The result of the shader on a Spanish moss model is seen in the following sixteen images in Figure 4.1 and Figure 4.2. These images are taken from a turntable that was rendered using the shader.

The next set of images (Figure 4.3) reveal how displacement depth affects the shader. The displacement depth controls how strong the displacement is. It tells the shader how extreme the depths and heights of the bumpiness of the displacement map. In image (a) of Figure 4.3, the displacement depth is set to 0.0 which will result in no displacement on the model. The other three images have a displacement depth that increment gradually.

Next, FBM noise surface frequency and its effects on the model is shown in Figure 4.4. The FBM noise surface frequency controls the frequency which is how many waves pass in a certain parameter. We use this FBM noise to create a pattern of noise between two colors in surface color. Although the change is subtle, the images in Figure 4.4 show the changes of the FBM noise surface frequency.

Next, the images in Figure 4.5 show the variation in fjump in the Fractal Brownian Motion noise function. The fjump is predefined number that is multiplied with and changed each time the for loop is executed in order to alter the noise pattern in the FBM noise function. The images in the figure (Figure 4.5) show the subtle differences in the fjump value change.

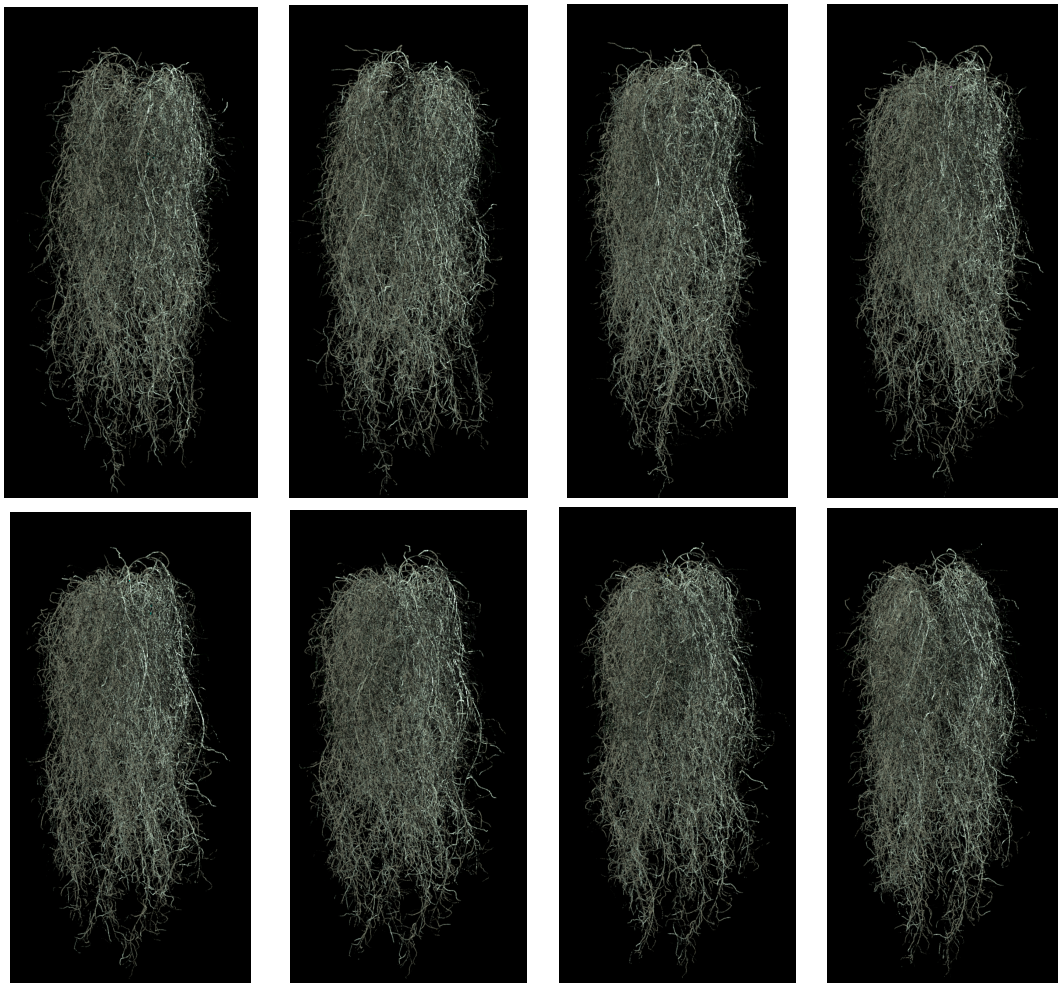


Figure 4.1: Selected Frames from Turntable of Spanish Moss

(a) (b)

Figure 4.2: Displacement depth which controls the strength of the displacement bumpiness. (a) 0.0, (b) 10.0

(a) (b)

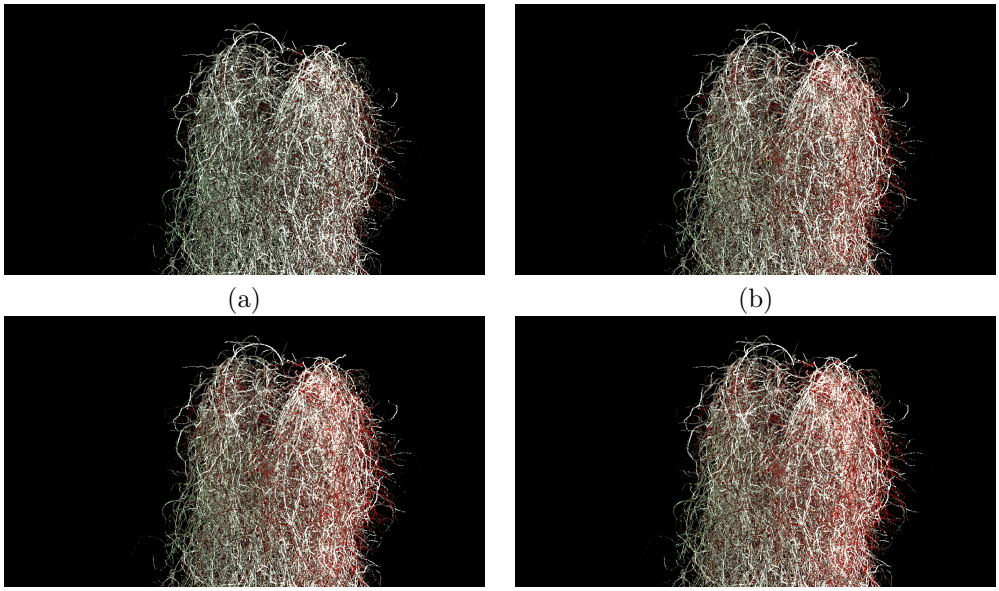
Figure 4.3: FBM noise surface frequency which affects the pattern of the noise of the two diffuse colors. (a) 0.0,(d) 10.0

(a) (b)

Figure 4.4: Fjump which affects the fractal brownian motion noise. (a) 0.0, (b) 3.2

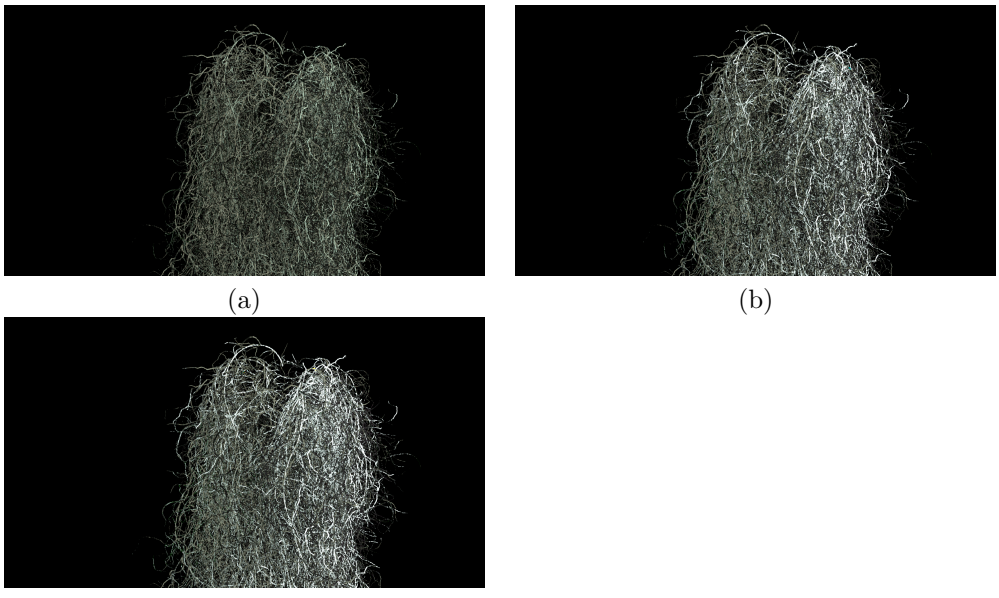
(a) (b)

Figure 4.5: FBM noise Roughness which affects the strength of the mixture of the two diffuse colors. (a) 0.0, (b) 1.0



(a) (b) (c) (d)

Figure 4.6: Kss value which controls how strong the subsurface scattering is and subsurface scattering selection of the color red for better visibility. (a) 1.0, (b) 2.0, (c) 4.0, (d) 5.0



(a) (b) (c)

Figure 4.7: Specular roughness which controls how broad the specular highlights are. (a) 0.0, (b) 0.5, (c) 1.0

FBM noise Roughness as seen in Figure 4.6 shows the mixing of two colors that are generated into a noise pattern by the Fractal Brownian Noise function. The higher the FBM noise Roughness, the stronger the secondary color shows. In image (a) and (b), the FBM noise Roughness is at a lower value and the first color, which is a green tinted grey, is strongly visible. But as the FBM noise Roughness is set to higher number, as in image (c) and (d), the secondary orange tinted grey is more visibly seen. This cause more color variation in the surface color which results in a more naturally colored moss.

In Figure 4.7, a few of the parameters have been altered to show the subsurface scattering better, The Kss value changes but the subsurface scattering color is changed to red and the lights are a bit brighter to better see the change. The higher the Kss value, the stronger the subsurface scattering shows and it over takes the surface color as see in image (d).

Finally, the images in Figure 4.8 are examples of the way in which specular roughness alter the appearance of a shader. In image (a), the specular roughness is set to the value of 0.0. No specular highlights are visible. As the values are increased in images (b) and (c), the speular highlight becaomes more visible and broader in size.

In conclusion, these images not only show the final version of the shader on a Spanism moss model, it also shows how the attributes can be altered. For example, if an artist was following an artistic vision for other wordly, alien Spanish moss, the artist could still use this shader. The shader would have the physical properties of the hair shader such as BRDF, subsurface scattering, etc., but the color, frequency, and other attributes can be altered in order to meet the artistic vision. The colors of the moss can be changed as well as the displacement depth and the other attributes in order to fulfill the art design of this new breed of Spanish moss. This shader is versatile and available for a great deal of adaption and exploration.

Chapter 5

Conclusions and Discussion

Although many may think that models and shaders can only be used for feature length computer animated films, it is actually much more versatile than for solely that purpose. This shader for Spanish moss can be used for television, films, and games. Additionally, this shader can be used for set extensions for live action films as well. In current and future films, the line between live action films and computer generated films is becoming more and more blurred. Modern audiences accept that the many effects such as explosions, fire, giant green monsters such as the Hulk (Figure 5.1) are created using computer generated images and effects rather practical effects such actual fires and wrestlers in green paint. These effects now can also be artistically driven and be altered to follow a visual style. For example, films are no longer limited to orange fires, but they can changed into blue flames that shift to purple or whatever color scheme the film needs. Water waves can be guided into a certain direction. The computer generated waves can be artistically driven to move in certain patterns and ways rather than just throwing physical water against something and hoping for the best result. The water can be created to beat against a certain object. An example of this is the water beating against the basket in Figure 5.3. This water is designed to move iin designed and controlled currents as the wizard rides the hot air balloon basket down the computer generated rapids. The use of green screen and compositing is becoming more used in filmmaking as the technology progresses. In fact, many films use live action human actors, but most of the sets are created digitally with computer generated models, surfacing, and animation. This allows for artists to create a new and unique world that they have absolute control over. It does not matter if the weather is overcast or if the filmmakers have permission to film at a particular location. The

computer artist can create the environments, props, and additional materials that can be altered and locked once a desired style has been achieved. An great example of this invention of a stylized alternate world is in "Alice in Wonderland" (Figure 5.2). The actors are filmed on a green screen and blue screen set and they are then composited into a computer generated world. A few examples of films that use this method are "The Avengers (2012)", "Oz, the Great and Powerful (2013)", "Alice in Wonderland (2010)", "Avatar (2009)", "The Jungle Book (2016)", and many others.

The use of computer generated imagery in live action films can be helpful in other situations rather that just creating entire backgrounds or monsters or special effects. It can be used to create cohesion in a film. In most cases, movies are not filmed in only one location. Often times, there multiple locations that can be in different parts of a city or in entirely different countries. Although the filming may take place in two entirely different countries, the film's setting may be designed only to be in one location. Filmmakers have the ability to use computer generated imagery to add to or take away from aspects of the filming locations in order for the film to read as being set in one location.

Spanish moss is an environmental piece that can subtly notify the audience of the film's location. Since Spanish moss is found in the swampy, lowcountry areas of the South, its presence leads the audience to assume that the setting is more likely to be in South Carolina rather than England or some other place where Spanish moss does not grow. Spanish moss can also be used to tell the audience that the film's setting takes place in the old gothic South from long ago. Or pehaps, that the film takes place on an old plantation in 1860's Georgia. A few films take advantage of Spanish moss's ability to signal a geographic location and incorporate it into the environment of their films. An example of this is from "Forrest Gump" (Figure 5.4). "Forrest Gump" was primarily filmed in South Carolina and Georgia, but the film's setting is in a fictional town in Alabama, which is a Southern state in the United States. This use of Spanish moss helps the audience to place this town in the deep South of the United States. The moss is hanging from the trees behind Forest as he is talking while sitting on a bench. Because "Django Unchained(Figure 5.5)" deals with slavery and racism, it's setting is in the American South. Spanish moss is often associated with large oak trees shading mansions on old plantations. Here, the moss is used to accentuate that. Additionally, "The Patriot (Figure 5.6)" was partially based in South Carolina for the film and was also shot in South Carolina. The moss is subtle in the trees behind the British soldiers, but the moss adds the southern, rural components to the mise-en-scne of the film. Finally, the film, "The Notebook(Figure

Figure 5.1: The Use of Green Screen and Compositing Example in "The Avengers(2012)" [14]

Figure 5.2: The Use of Green Screen and Compositing Example in "Oz the Great and Powerful(2013)" [14]

Figure 5.3: The Use of Green Screen and Compositing Example in "Alice in Wonderland(2010)" [14]

Figure 5.4: Still from "Forrest Gump(1994)" [15]

Figure 5.5: Still from "Django Unchained(2012)" [16]

5.7)," was based in South Carolina and filmed in parts of the lowcountry of that state. This shot was filmed in Cypress Gardens of Moncks Corner, South Carolina. Although they are far into the background, the Spanish moss hanging from the cypress trees in the swamp sets the environment of a Southern American swampland. The use of moss in these films set the locations and helps to inform the audience of that information in a cinematic way.

Since Spanish moss can set the location of the film, the film crew may want to add Spanish moss to trees for the film. While one can hire a team to obtain moss and layer it into the tree branches, one may not want to pursue this route. This Spanish moss shader can be used to add

Figure 5.6: Still from “The Patriot(2000)” [17]

Figure 5.7: Still from “The Notebook(2004)” [18]

Spanish moss using compositing if the filming location does not have Spanish moss, but the film calls for this plant in the environment. Compositing is the process of combining two frames or images together in order to create a new frame or image. An artist can composite frames that include the Spanish moss on top of the live action film that includes the actors and environment. The Spanish moss can be rendered using similar lighting as the lighting in the live action film so that the moss will appear naturally growing in the trees. Using compositing, Spanish moss can be added to trees in post production in order to help establish the film's location.

In conclusion, the goal of this shader is to provide a procedural shader for Spanish moss. By using different attributes, physical attributes of Spanish moss are translated into a surfacing shader that an artist can use in order to create photorealistic Spanish moss in a computer graphics world.

Bibliography

[1] Plant Divisions: Mosses, Liverworts and Hornworts. Tentative Plant Scientist, 20 Sept. 2015, tentativeplantscientist.wordpress.com/2013/04/02/plant-divisions-mosses-liverworts-and-hornworts/. Accessed 9 Mar. 2017.

[2] Suzanne Benton. *Spanish Moss in Sunlight*. 2017

[3] Upstill, Steve. The RenderMan Companion: a Programmer's Guide to Realistic Computer Graphics. Boston, Addison-Wesley, 2005.

[4] Cortes, Rudy, and Saty Raghavachary. The RenderMan: Shading Language Guide. Boston, MA, Thomson Course Technology, 2008.

[5] Light Scattering from Human Hair Fibers. *Light Scattering from Human Hair Fibers*, www.cs.cornell.edu/ srm/publications/SG03-hair-abstract.html. Accessed 10 Mar. 2017.

[6] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation. BRDF Measurements with Robots, www.iosb.fraunhofer.de/servlet/is/14836/. Accessed 10 Mar. 2017.

[7] BSDF. BSDF Lumerical Knowledge Base, kb.lumerical.com/en/particle_scattering_bsdf.html. Accessed 10 Mar. 2017.

[8] Image Synthesis Techniques. CS 348B, graphics.stanford.edu/courses/cs348b-competition/cs348b-02/tam/. Accessed 10 Mar. 2017.

[9] Subsurface Scattering - CRYENGINE Manual. Documentation, docs.cryengine.com/display/SDKDOC2/Subsurface+Scattering. Accessed 10 Mar. 2017.

[10] Pixar. *Shading a Coral Reef. Pixar's RenderMan*, renderman.pixar.com/view/shading-a-coral-reef. Accessed 9 Mar. 2017.

[11] Coral Polyps. *Coral Reef Alliance*, coral.org/coral-reefs-101/coral-reef-ecology/coral-polyps/. Accessed 10 Mar. 2017.

[12] Great Barrier Reef. *Pixar Wiki*, pixar.wikia.com/wiki/Great_Barrier_Reef. Accessed 10 Mar. 2017.

[13] Pharr, Matt, et al. Physically Based Rendering: from Theory to Implementation. Cambridge, MA, Elsevier/Morgan Kaufmann Publishers, 2017.

[14] 46 Famous Movie Scenes Before And After Special Effects. *Digital Synopsis*, digitalsynopsis.com/design/movies-before-after-green-screen-cgi/. Accessed 11 Mar. 2017.

[15] Bianco, Antonella. Uomini Pi Stupidi Delle Donne? Quasi Scientifico, Ecco Il Motivo. *ItaliaOra*, 15 Oct. 2015, italiaora.retenews24.it/uomini-piu-stupidi-delle-donne-e-quasi-scientifico/. Accessed 12 Mar. 2017.

[16] Imgur. Tarantino's Signature 'Trunk Shot' in Django Unchained. *Imgur*, 22 Jan. 2013, imgur.com/gallery/l8Jy692. Accessed 12 Mar. 2017.

[17] Jasoncherry2002. The Patriot Movie Trailer. YouTube, *YouTube*, 19 Jan. 2010, www.youtube.com/watch?v=_comGBmnYew. Accessed 12 Mar. 2017.

[18] Stefknbauwens. The Notebook - Trailer. YouTube, *YouTube*, 17 July 2012, www.youtube.com/watch?v=4M7LIcH8C9U. Accessed 12 Mar. 2017.

[19] Peterson, J. Scott. Spanish Moss. plants.usda.gov/plantguide/pdf/cs_tius.pdf. Accessed 9 Mar. 2017.