A SHADER BASED APPROACH TO PAINTERLY RENDERING

A Thesis

by

KAUSHIK PAL

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

MASTER OF SCIENCE

August 2004

Major Subject: Visualization Sciences

A SHADER BASED APPROACH TO PAINTERLY RENDERING

A Thesis

by

KAUSHIK PAL

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Karen Hillier (Chair of Committee)	Donald House (Member)
Donna Hajash (Member)	Phillip Tabb (Head of Department)

August 2004

Major Subject: Visualization Sciences

ABSTRACT

A Shader Based Approach to Painterly Rendering. (August 2004)

Kaushik Pal, B.Arch., Birla Institute of Technology

Chair of Advisory Committee: Prof. Karen Hillier

The purpose of this thesis is to develop a texture-based painterly shader that would render computer generated objects or scenes with strokes that are visually similar to paint media like watercolor, oil paint or dry media such as crayons, chalk, et cetera. This method would need an input scene in the form of three dimensional polygonal or NURBS meshes. While the structure of the meshes and the lighting in the scene would both play a crucial role in the final appearance of the scene, the painterly look will be imparted through a shader. This method, therefore, is essentially a rendering technique. Several modifiable parameters in the shader gives the user artistic freedom while overall introducing some amount of automation in the painterly rendering process.

To my parents

ACKNOWLEDGMENTS

I would like to take a moment to thank my committee chair, Prof. Karen Hillier, for her knowledge and guidance which helped me to accomplish this thesis and led me through the difficulties in this research. Special thanks to Dr. Donald House who not only served on my committee, but also helped me to grow and to learn in the lab. I would also like to thank my other committee member, Donna Hajash, for her enthusiasm and encouragement. Also, I would not be able to finish this thesis without a very special person, Seema, who gave me hope, support, strength, friendship and love. Above all I would like to thank my parents for their encouragement, understanding and unconditional love which has helped me all through my life.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

I.1. Motivation

I.1.1. Traditional Painting

1. **Styles and Media** A painting captures the essence of the subject as the painter sees it. Through color and the nature of brush strokes used by the painter the painting can evoke many emotions. A painting, unlike a photograph is not an exact depiction of reality but is rather an interpretation of it. Even though the subject of a painting may be real life objects, the painter presents an abstraction of it, thus involving the viewer in the interpretive process. By using various shapes, sizes and density of brush-strokes, a painter can establish the interaction of light with a surface and the texture of a surface. Figure 1 shows how the different strokes work for the floorboards, the

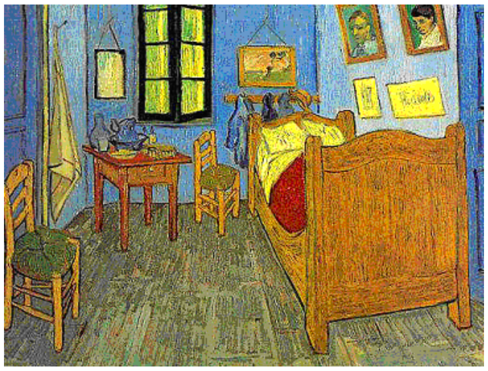
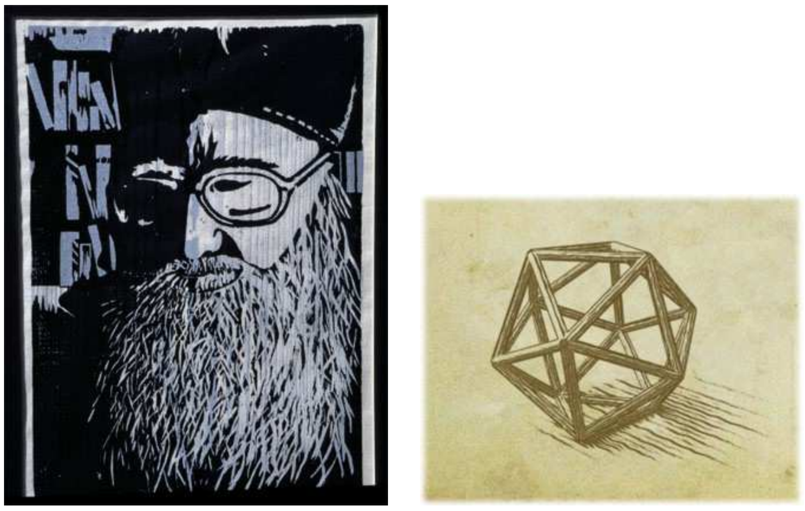


Fig. 1. Vincent Van Gogh's Room at Arles [8].

bed and the walls creating a textural distinction between these objects. In computer

The journal model is *IEEE Transactions on Visualization and Computer Graphics.*

graphics (CG) there are easily automated processes to depict every object in the scene in fine detail. Traditionally, computer graphics uses the physical properties of color, light and perspective to generate realistic images. These images are thus more photographic or photoreal in nature. Techniques used in CG that allows artists to create images that do not necessarily conform to the physical laws are called non photorealistic rendering techniques. Non-photorealism by definition encompasses all visual styles that do not strive to represent subjects as they would look in a photograph or appear to the human eye. Painterly rendering is a subset of non photorealistic rendering. Other visual styles that fall into the non-photoreal category include illustration (pen and ink, technical, wood block, etching) and cartoon shading.



(a) Traditional art (b) CG

Fig. 2. Woodblock printing in traditional art and in CG [9, 1].

An example of woodblock developed as a plugin by Expressive Studios for Maya is seen in Figure 2. The traditional technique, seen in Figure 2 serves as the inspiration for creating an identical look in CG.

While photorealism can be desirable for many purposes such as achieving a cinematic effect or incorporating a CG object seamlessly into a real life scene, non-photorealism has many other important advantages in the artistic process of creating effective visuals. The general audience expects realistically rendered characters to behave realistically. Therefore, to express ideas beyond the physical and logical norm, in a way that is acceptable for general audience, non-photorealism could be used. Through the ages different styles or schools of painting have developed. The evolution of a particular school of thought may have been influenced by the several factors, society, religion and politics being the main influences. The differences in these styles are always manifested visually through subject, color or texture. Apart from the artistic movement it represents every painting owes its visual uniqueness to the individuality of the artist's expression. Interestingly, contrary to the ideas of non-photorealism or more specifically the ”painterly” look, a movement called photorealism exists in traditional painting as well.

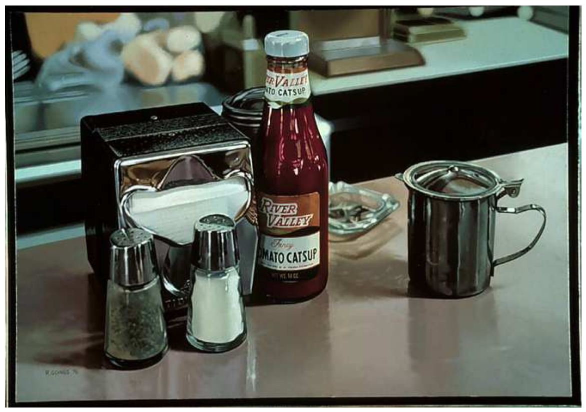


Fig. 3. Ralph Going's River Valley still life [8].

It is a figurative movement that emerged in the United States and Britain in the late 1960s and 1970s. The subject matter, usually everyday scenes, is portrayed in an extremely detailed, exacting style. It is also called superrealism, especially when referring to sculpture. An example of realism in painting is shown in Figure 3. In contrast, an artistic movement that largely influences this thesis in the visual context is Impressionism.

2. **Impressionism**

French Impressionnisme, was major movement, first in painting and later in music. Impressionism developed chiefly in France during the late 19th and early 20th centuries. Impressionist painting comprises the work produced between about 1867 and 1886 by a group of artists who shared a set of related approaches and techniques.

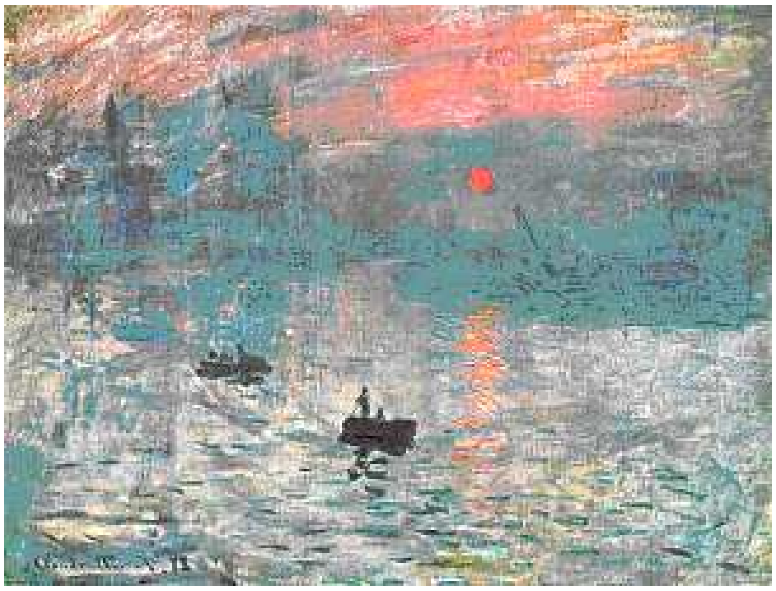


Fig. 4. Claude Monet’s Impression: Soleil Levant [8].

The most conspicuous characteristic of Impressionism was an attempt to accurately and objectively record visual reality in terms of transient effects of light and color.The artist often used bold brush strokes and color to capture the impression of light often in an out door environment. Figure 4 shows bold brushwork the resulting sense of abstraction of a sunrise by Monet.

3. **Paint media** The physical properties of artistic media also significantly contributes to the visual style of the painting. Fluidity/ viscosity, interaction with light, the drying time of the paint medium and the opacity of the medium are some of the physical properties of paint that not only creates unique visual styles but also influence the painting technique.

4. **Time Based Media** ***Fantasia 2000***- A Walt Disney production, directed by James Algar, [2] the film featured ground breaking work in animation. The feature was created using a combination of several techniques from hand drawn frames to computer generated effects. It was an ambitious Walt Disney project that combined several styles of animation that played seamlessly to a classical music soundtrack.

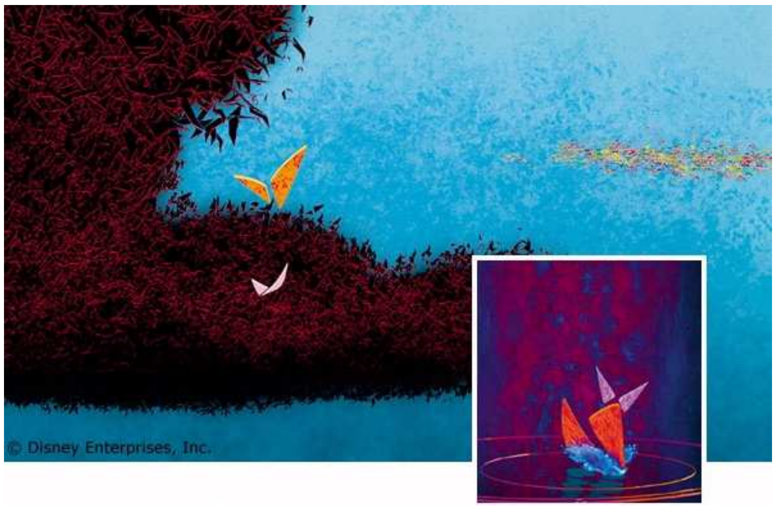
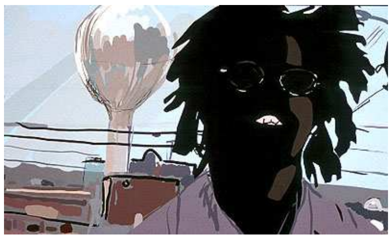
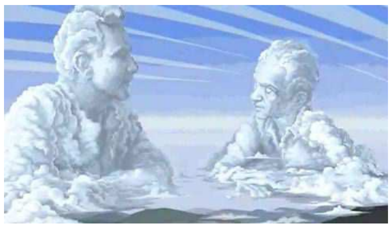


Fig. 5. A frame from ’fantasia 2000’ in the segment ’Symphony no. 5’ [2].

Figure 5 shows a frame from the first segment of the film. It is an example of an abstract rendering of an abstract musical piece, Beethoven’s ”Symphony No. 5”. To give the visuals a graphical look, textures were made from scanned pastels. The shapes were flattened onto planes, outlined procedurally and layered in Z depth.

Waking Life- directed by Richard Linklater [15] The visuals of Waking Life are created by animators painting over actual video footage. The painting was done frame by frame, with some key framing and computer interpolation, which imparts a dynamic character to the animation. The director chose to portray ideas that may be entirely a parts of one's mind. This called for a set of visual styles that would better interlock the ideas and the imagery. The visual styles used in the film ranged from cubist, oil painting to a comic book look. The stylized visuals served to extend the film's metaphysical state. Figure 6 shows some of the visual style used in the film. Fig 6A diplays a flat shaded and outlined comic book look, Fig 6B and Fig 6C are more painterly in style.


(a) comic book style


(b) painterly style

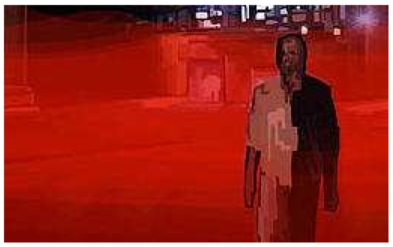

(c) painterly style

Fig. 6. Three frames from the film 'Waking Life' [15].

What Dreams May Come- directed by Vincent Ward.[13] The painted world sequences in this film uses painterly particle effects to create an environment like an impressionist painting. The digitally created 'paint' is made to interact with the principal character in the film who finds himself 'inside' the painting. The movement of the particles in a viscous fluid-like fashion help in making the paint appear wet. Figure 7 shows a frame from the impressionistic painting sequence of the film.

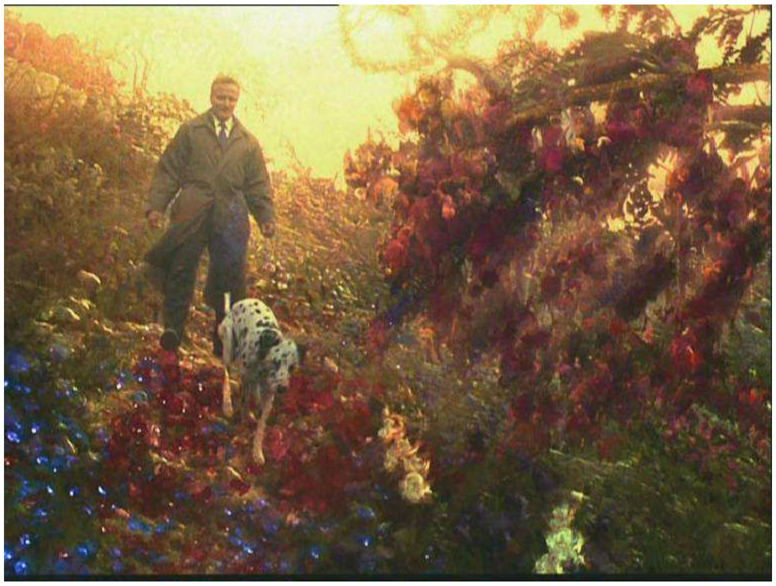


Fig. 7. A frame from the film ’What Dreams May Come’ showing the impressionist world created in the film [13].

I.2. Characteristics of Non-Photorealistic Imagery

Although, a broad range of visual styles may fall in the domain of non photorealism it is possible to identify certain salient features that non photorealistic imagery have. The following are some of the characteristics seen in many non photorealistic styles.

1. **No sharply defined object boundaries**

   objects are defined with discreet strokes which cohesively define the shape of the object. The artist manipulates the size, shape and density of strokes creating a subject that is more implied rather than explicit.

2. **Level of Detail** The number or density of strokes control the amount of detail desired in an object. Some painters use an underpainting with large brush strokes and then bring out the details with smaller strokes. Level of detail is also dependent on the

distance of an object. Distant objects are rendered with fewer details than objects that are closer. This technique also serves to provide depth in a scene.

3. **Spatio-temporal coherence** Temporal coherence is not a problem for still imagery but is important in animation. Frame by frame coherence is achieved when the media moves with the object. Without tracking, objects appear to be moving behind the media, this is known as the ”shower door” problem.

4. **Entropy** Although some degree of coherence in animation is desired, perfect coherence would be defeating the purpose as hand crafted media is imperfect. The imperfections handcrafted media impart the subject with an energy that is one of its most salient aspects and is therefore desirable that it is retained in computer generated animation. However if the frequency of these imperfections are too great over frames then it just produce just distracting noise. The key is to use these imperfections as features to generate a particular stylization, mood, texture or energy at the artist’s discretion.

5. **Color and light** color and light in non-photorealistic rendering can be used in abstract ways. The use of complementary colors is common in traditional art where the underpainting in a complementary color helps lift the subsequent layers. Light and shadow can often be depicted in nonphotorealism not just by increasing the shadowed region in value but it may be accomplished by shifting the hue often to a cooler color.

Problem Statement: The scope of this thesis may be defined based on characteristics of non-photorealism and the stylistic influences discussed above. The goal would be to simplify the rendering pipeline and to achieve a painterly effect over a minimum intermediate rendering steps. The painterly look will be captured in a generic way without attempting to simulate any particular medium with great precision.

CHAPTER II

BACKGROUND/PREVIOUS WORK

Non-photorealistic rendering is a rapidly growing subgenre of computer graphics. There is a trend among digital artists to imitate other media. Recent work on painterly rendering [10] , pen and ink illustrations [18], and watercolor [17] have shown interests in imitating other media. A subset of non-photrealistic research has been on synthesizing ”flat media”. These include canvas or paper based techniques, like drawing, painting, pen and ink illustrations. Techniques like color segmentation, limited palate and edge detection algorithms are a few of several ways to achieve painterly imagery. Several of these techniques can be applied to 2D as well as 3D input formats. However, the implementation of these techniques for a 2D input varies from that of 3D. Images (rendered frames, scanned photographs) and video footage (Image sequence) fall in the 2D input category. Various post processing techniques may be applied to achieve non photorealistic effects. 3D input scenes composed of computer generated geometry (polygons, NURBS et cetera). 3D objects can be rendered in non-photorealistic styles using a texture/shader driven approach or a force field/simulation driven particle system. There are various other methods but the above mentioned are the most commonly used. Subcategories of the non-photorealistic output can be but are not limited to any of the following more common visual styles:

1. **Illustration** (pen and ink, technical, wood block effect, etching)

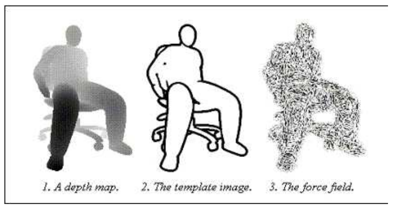
2. **Painterly** (Watercolor, Impressionist, other painting media)

An understanding of traditional flat media and their inherent properties is essential in trying to mimic them in the CG environment.

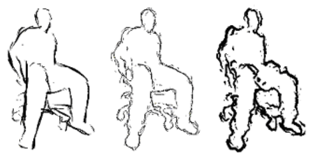
II.1. Previous Work Based on 3D Input Schemes

II.1.1. Particle System Based Rendering

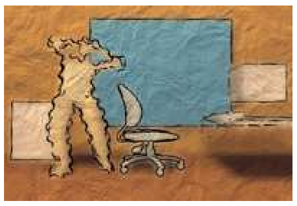
A particle system based work flow is used by some researchers to achieve non-photorealistic effects. A 3D input scene is generally used in this process. Particle systems offer the advantage of creating an approximation of the actual geometry in the scene. This gives very natural broken edges to the object which is a desirable property of painterly rendering. Also, particles can inherit brush images and their attributes to give brush stroke based painterly results.



(a) depth map, template and force field images used to drive the particles



(b) Various silhouettes created by particles

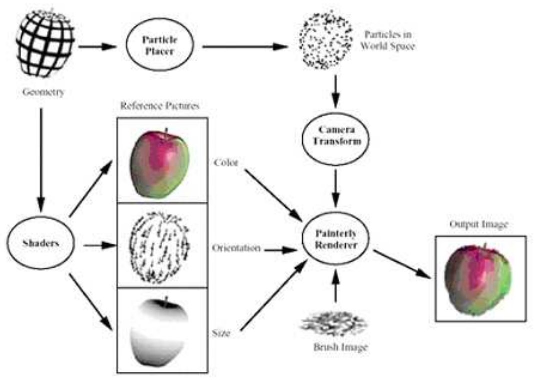


(c) Final Result

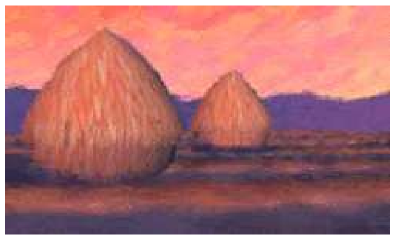
Fig. 8. 'Cassidy Curtis' Loose and Sketchy Animation [3].

Figure 8 is an example of Cassidy Curtis' work on loose and sketchy animations. He sought to demonstrate the power of a hand drawn sketch in conveying information and emotional states. The image in Figure 8 captures the energy of the artist's hand and since it produces an incomplete image it encourages the viewer to interpret the artist's idea and complete the picture by imagining the missing details. As shown in Figure 8, Curtis achieved the hand drawn sketchy effect using a particle driven system that is driven by a force field generated by a depth map of the CG object.

Barbara J. Meier's painterly rendering for animation is closely related to this thesis in



(a) Painterly rendering pipeline



(b) Final Result

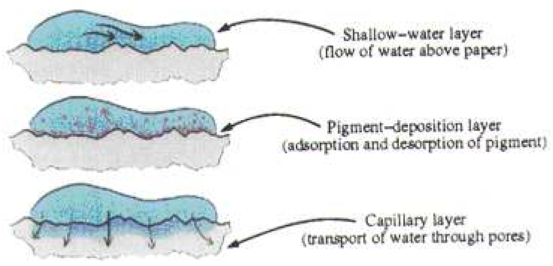
Fig. 9. Barbara J. Meier's Painterly Rendering for Animation [10].

terms of visual character. As in this thesis, her work is influenced by impressionist paintings. However, Meier's work differs in the technique used to get painterly effects. Meier, like Curtis uses a particle based approach. Figure 9 shows an example of the painterly rendering pipeline. The particle placer populates a surface with particles. The surface geometry is rendered using various shaders to create particle attribute reference pictures such as color, orientation and size. The particles, which are transformed into screen space, the reference pictures and the brush image are input to the painterly renderer. The renderer looks up brush stroke attributes in the reference pictures at the screen space location given by each particle's position and renders brush strokes that form the final image as seen in Figure 9.

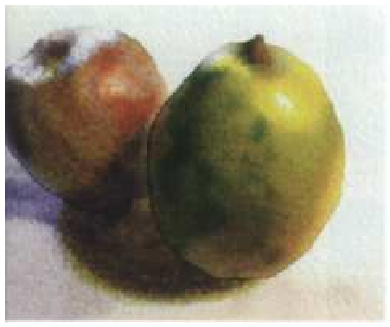
II.1.2. Simulation Based Painterly Rendering

An approach to creating likeness to traditional media in CG is to simulate that particular medium's physical properties. The algorithm is based on the actual properties of the medium with respect to its interaction with light, surface and pigment. This approach may often give a very accurate representation of the medium. Casidy J. Curtis, Sean E.

Anderson, Josua E. Seims, Kurt W. Fleischer and David H. Salesins' work on Computer Generated Watercolor used a fluid simulation approach. Figure 10a shows the interaction of water with paper and pigment was studied from a physically based stand point resulting in a realistic watercolor appearance as seen in Figure 10b.


Shallow–water layer
(flow of water above paper)
Pigment–deposition layer
(adsorption and desorption of pigment)
Capillary layer
(transport of water through pores)

(a) Water, pigment and paper interaction



(b) Final result

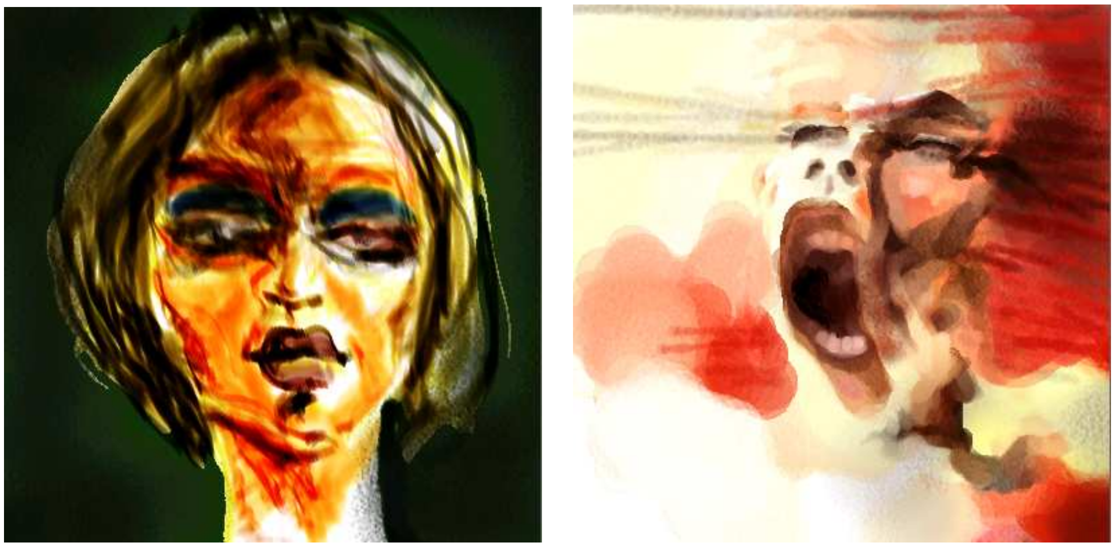
Fig. 10. Cassidy J. Curtis, Sean E. Anderson, Joshua E Sims, Kurt W. Fleischer and David H. Salesins' Computer Generated Water Color [17].

II.1.3. Commercial Software Packages

Several commmercially available sofware packages may be used for painterly rendering purposes. The majority of these are 2D image manipulation programs that give painterly rendering by applying particular filters to the 2D input image. Some commercial packages facilitate the creation of digital paintings. The artist can use the brushes provided in such packages to paint on a digital canvas using some input device such a wacom tablet or mouse.

1. **Fractal Painter** developed by Metacreations and later by Procreate, Fractal Painter 5 and Painter 7 have established themselves among the best image creation program for projecting natural media effects. It earned that reputation with its standard brushes such as chalk, crayon, pencil, charcoal and watercolor and oil paint. This unique

painting program allows one to create digital paintings with virtual brushes, pens, markers, canvases, and other types of art materials that act like their real-world design situations. Figure 11 shows some paintings done in Fractal Painter.

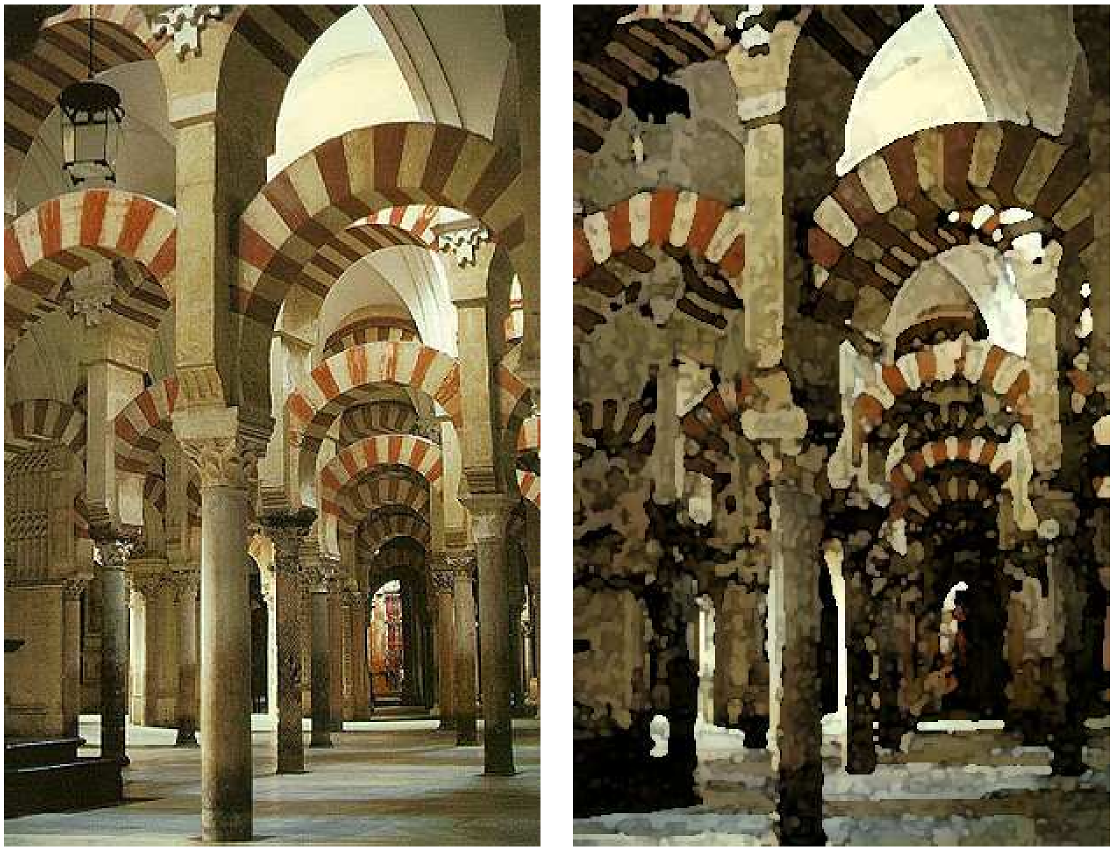


(a) marker brush (b) Watercolor brush

Fig. 11. Digital paintings with Fractal Painter.

2. **Adobe Photoshop** is a premier digital image editing, photo retouching, and color painting software with powerful features for graphic artists and designers. While Fractal Painter is a painting tool, Photoshop is primarily a tool for digital image editing allowing for both image renderings, editing and graphic manipulation. Photoshop's extensive bitmap manipulation tools make it the reigning supreme of image editing. Photoshop has several standard filters designed to mimic traditional media. Using these filters one can create many painterly effects such as watercolor, chalk and pen and ink. Figure 12 shows one such filter creating watercolor effect.

3. **Pixar's Renderman Utility** Renderman enables its shaders to output arbitrary vari-



(a) Original image (b) Image with Watercolor filter

Fig. 12. Image manipulation with Adobe Photoshop 7.

able values. This helps in the output of useful information like color maps, shadow maps, light maps and normal maps. This feature is particularly useful for non-photorealistic rendering, especially by the use of post processing. Thus, many artistic or painterly effects may be achieved by performing two dimensional operations on final images. Figure 13 shows an example for painterly rendering in renderman. By using the image containing normals shown in Figure 13C to set brush direction, and the original color map shown in Figure 13A to set the brush color, one can create an automated process that literally ”paints” by picking random places on the image, rotating a brush stroke image to the angle dictated by the normal, and drawing a stroke

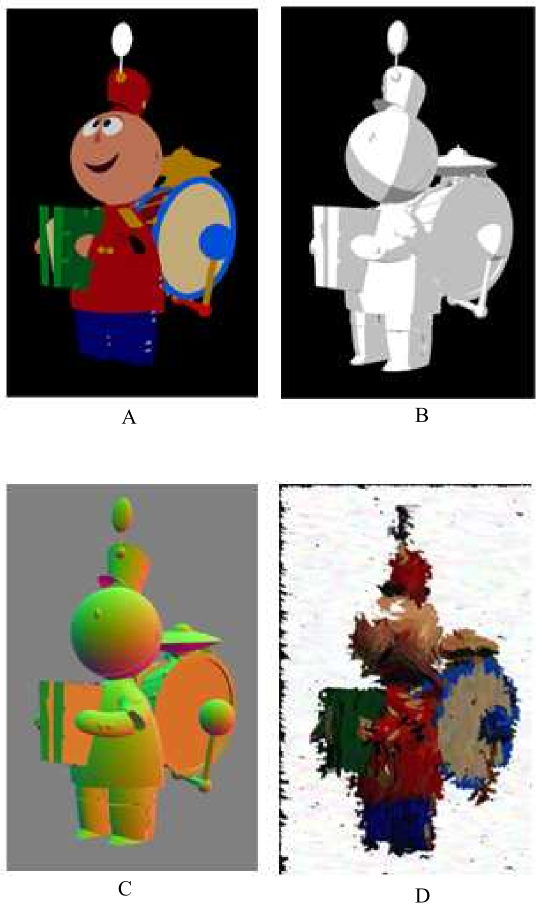


Fig. 13. A painterly rendering technique by Pixar's Renderman.

colored by the original input image. Some randomness has been added in the output image shown in Figure 13D to make the colors a bit more varied.

CHAPTER III

METHODOLOGY

This chapter will discuss the methodology to achieve the goal to create painterly images from a 3D scene with the geometry and a light setup. To approach the problem it will be necessary to define the required components in the input scene.

1. **Geometry:** the scene should have at least one polygonal or NURBS mesh. The painterly shader will be applied to this mesh.

2. **Lights:** the scene should have at least one light. The shader will color the object based on the way it is lit in the scene.

3. **UV mapping:** It may be necessary to UV map any non-primitive polygonal object. Mapping is important for the shader texture to wrap over the object correctly. The type of mapping (spherical, cylindrical, planer) used would depend on the shape of the object and also on the desired final look.

We will also formulate certain criteria from the usability point of view.

1. There should be minimum user intervention. The process should be almost automatic and over a minimum number of intermediate passes.

2. The user should, however, have artistic control over the images. In this case, artistic control will be in terms of color, texture and light.

The main steps in creating the nonphotorealistic scene are the following.

1. **Modeling:** Creating geometry for all elements that are in the scene. The models may be nurbs, polygons or subdivision surfaces. One of the main considerations during

modeling is the scale and orientations of the isoparms of the model, as the final texture placement will depend largely the position and orientation of the isoparms. Figure 14 shows the change in brush stroke orientation with the change in isoparm orientation on the geometry.

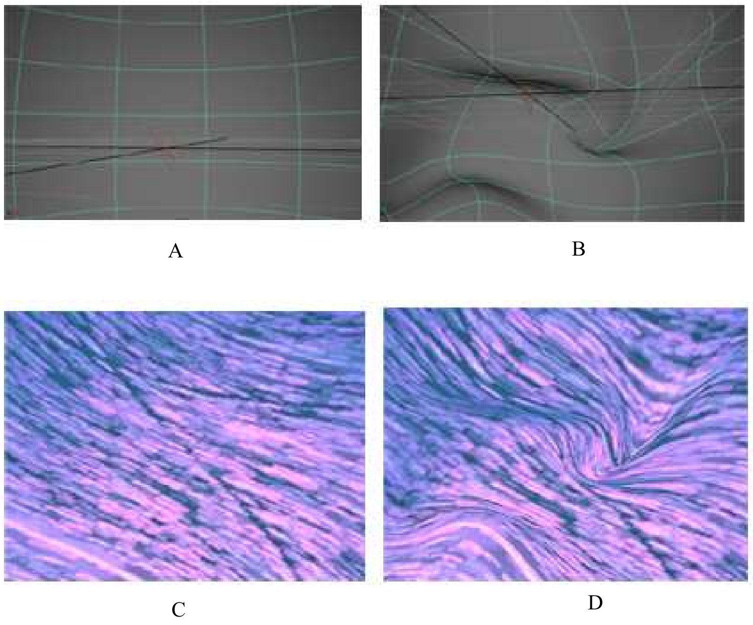
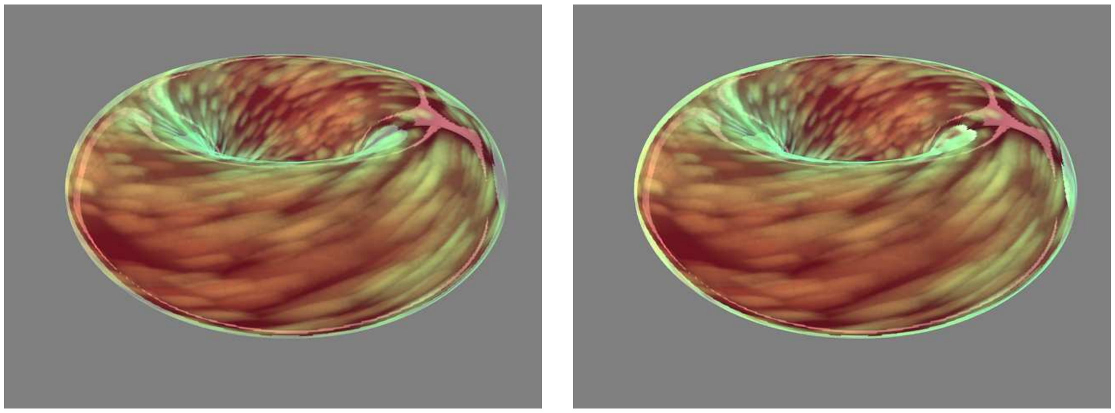


Fig. 14. Regular and deformed isoparms.

The user may also decide whether to use double sided geometry (geometry in which faces with both front and back facing normals will get rendered) or just front facing (faces with back facing normals are not rendered) objects. Using double sided objects make for denser brushstrokes and firmer edges while single sided objects objects will give more discontinuous edges when the shader is applied. Figure 15 show that single sided objects give a better impression of having broken edges.

2. **Lighting** The positions of lights in the scene will determine the final look of the ren-



(a) single sided torus (b) double sided torus

Fig. 15. Differences in rendering single and double sided objects 7.

ders in a significant way. Light direction will not only determine the surface properties of an object in terms of color but also whether the surface reflects or transmits light. Light types may be directional, spot, point, area or ambient.

Accounting for transmitted light is important for watercolor or other transparent media effects. Figure 16A shows a torus with a light positioned in front of it and directed at it. In this case the camera just catches the reflected light. Figure 16B shows the torus with the light behind it and directed at it. In this case the camera just catches the transmitted light. Figure 16C uses both the above described lights and therefore has both reflected and transmitted light.

3. **UV mapping:** This is necessary for the texture to map properly on the object. Texture on unmapped objects may be stretched or squashed. A polygonal object can be UV mapped in using planar projection, cylindrical projection or spherical projection. The way the texture wraps on the object will depend on the mapping technique used.

   Figure 17A shows the model with no uv mapping. notice how no brush strokes are seen in this. This is because the model's default uv do not fall correctly on the texture space of the shader. Figure 17B,C and D show the model as the brush strokes would

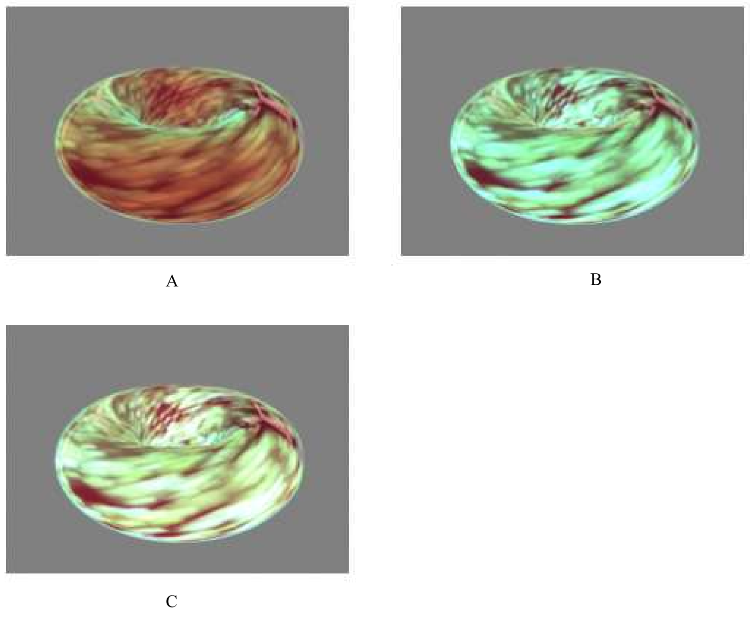


Fig. 16. Object appearance with reflected light, transmitted light and a combination of both reflected and transmitted light.

be seen with cylindrical, spherical and planer uv maps.

4. **Shading:** The shader will be written using renderman shading language. The compiled shader (in .slo format) can be imported to a slim pallete in the renderman MTOR interface.

   Conventional oil painting process-

   Conventionally the oil painting process involves the application of successive layers of paint to create the final result. Often an underpainting is applied as one of the first layers. The purpose of the underpainting is to block in the main masses in the painting and to also define an atmosphere through its color and tone. Subsequent layers are applied to enhance middle tone and details and the final layers add highlights and more details.

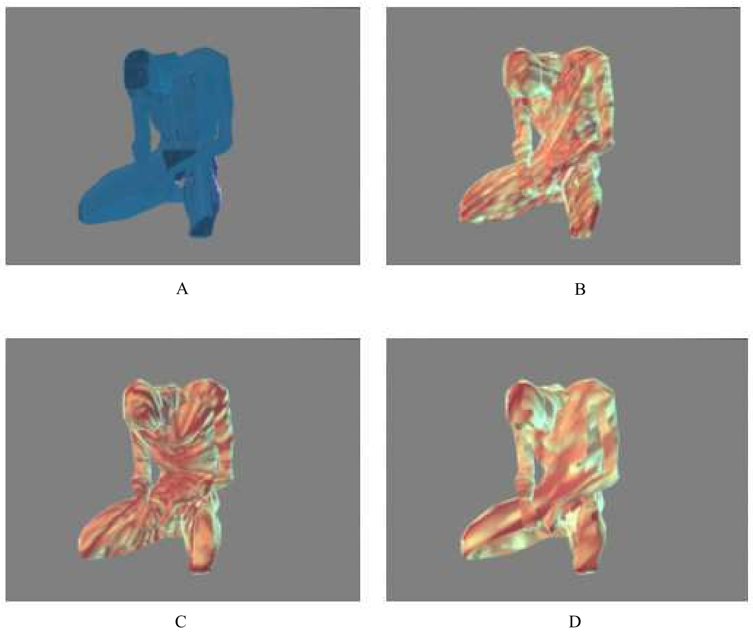


Fig. 17. Objects shown with different kinds of uv mapping.

Shader strategy- Much like the conventional oil painting method as seen in Figure 18, the shader will be created with layers of brush strokes. The first layer of the shader is the underpainting and subsequent layers form the overpainting.

Figure 19A shows the model in the scene to be shaded. Figure 19B shows the result after the underpainting and the outline layers. Although the basic shape of the object gets defined at this stage there is no light information. This stage can be likened to the underpainting stage in conventional oil painting. Figure 19C and D shows the object with successive layers of brush strokes. As with conventional oil painting the lighting gets more evident in this stage.

The total number of layers may be defined by the user. The sizes of the individual brush strokes increase with the number of layers and the level of detail increases. Examples of brush stroke images are seen in Figure 21. The shader reads in three

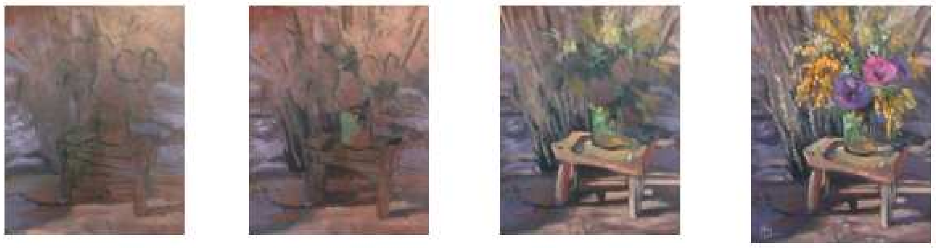


Fig. 18. The oil painting process [14]

’brush images’ that are sampled by the shader randomly. Each layer creates a texture with the randomly chosen stroke image. For texture creation the brush stroke is tiled repeatedly in the texture coordinate space (s, t coordinates). The frequency of the tiling depends on the layer number. For example in a shader with ten layers, the brush is tiles once for the first layer, twice over s and twice over t for the second layer and so on. Thus, for the tenth layer the brush will be tiled ten times on the s axis and ten times on the t axis. Each layer computes a color and opacity that is used for compositing it with successive layers. The brush strokes are placed on the layer’s texture according to randomly generated texture coordinates, random scaling and rotation values. The final texture for the object is the composite texture from all the layers. For the final opacity the shader uses the product of the brush texture opacity and the underpainting opacity. Many traditional artists tend to outline objects they want to emphasize in their paintings. The shader too has this functionality. Outlines or silhouette edges are determined by calculating the dot (inner) product of the view vector from the camera and the point on the object being shaded. Since the inner product of the two vectors give the cosine of the angle between the vectors, a simple threshold value for the inner product can be set for the outline. The value can be decreased to make the outline thicker and increased to make it thinner. In the shader's outline calculation routine some noise has been added to the inner product threshold to make the outlines broken and more painterly. The outline is calculated with a color and an opacity. It is composited with the other layers.

Coloring the brush strokes- The "base color" of the strokes is user defined. The user also defines a shadow color and a diffuse color. The color is multiplies with the grey scale brush image before it is committed to the tiled layered texture. Since the color operation is done before the brush is made into a tiled texture the texture is colored with discreetly colored brush strokes.

In traditional painting artists often convey a sense of distance using not only by reducing the level of stroke detail but also (as seen commonly in landscape paintings) by reducing the color detail of objects that are further away. Figure 20 shows examples Monet's work that create not only the illusion of distance but also makes for atmospheric effects. The shader handles this issue of imparting depth by calculating the length of the vector from the camera to the point being shaded in the scene and using that distance to shift the colors to an user defined value.

Brush Texture specifications- The brush images can be created in several ways. Any 2D paint program may be uses to create a variety of brush shapes. The final appearance of the scene is largely dependent on the method chosen to draw and sample the strokes. The user may provide the shader with up to three different brush images to be accessed randomly by the shader. These brush images may be made very different or very similar from each other depending on the stroke variation desired in the final image.

The stroke image file is read just to define the shape of the image and all color information is multiplied later to the image. The strokes can be a grey scale image that give the alpha information for the stroke. Thus the desired shape of the stroke must be in white for a stroke with hundred percent opacity or shades of grey for intermediate opacities. Regions of the image that are not part of the stroke, shape must be left black, making those regions totally transparent.

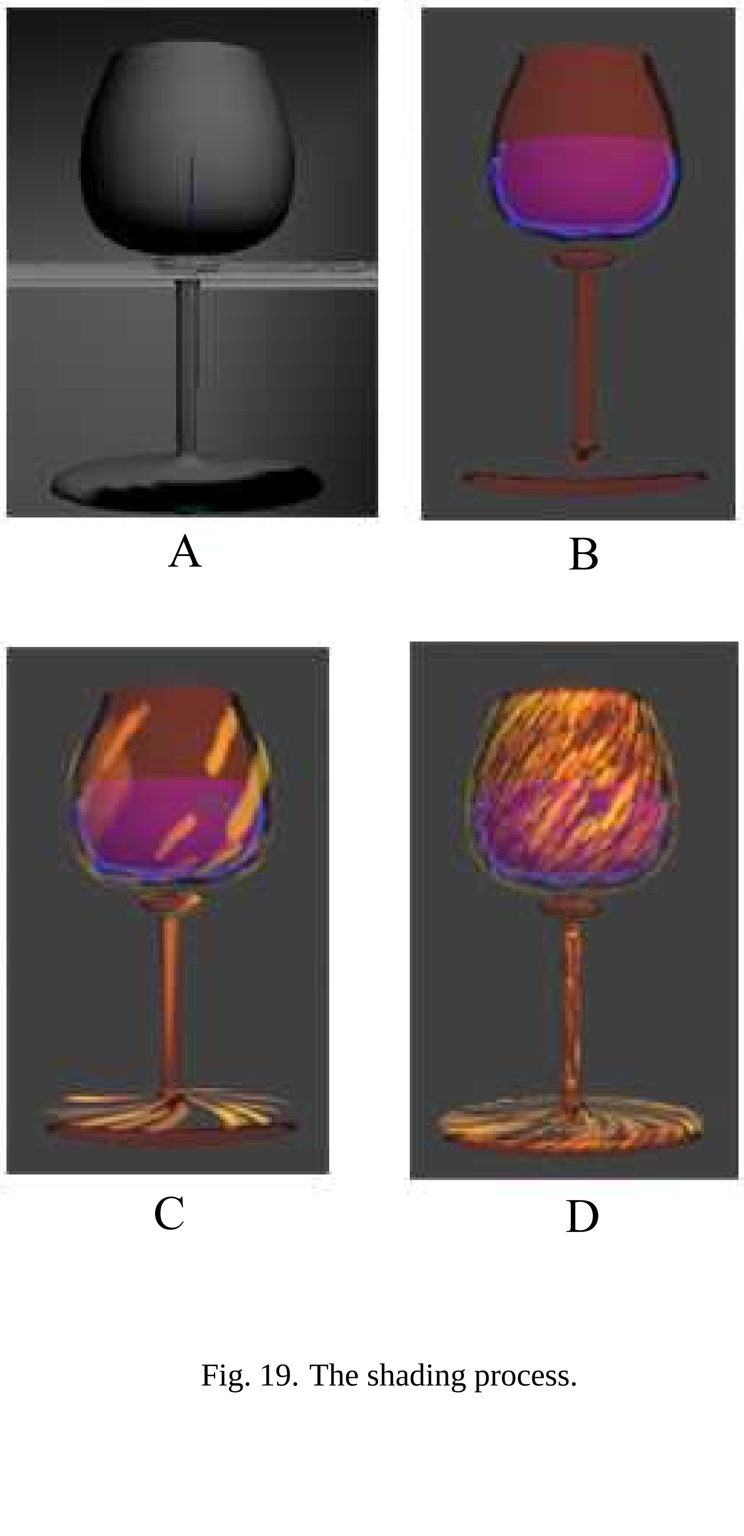
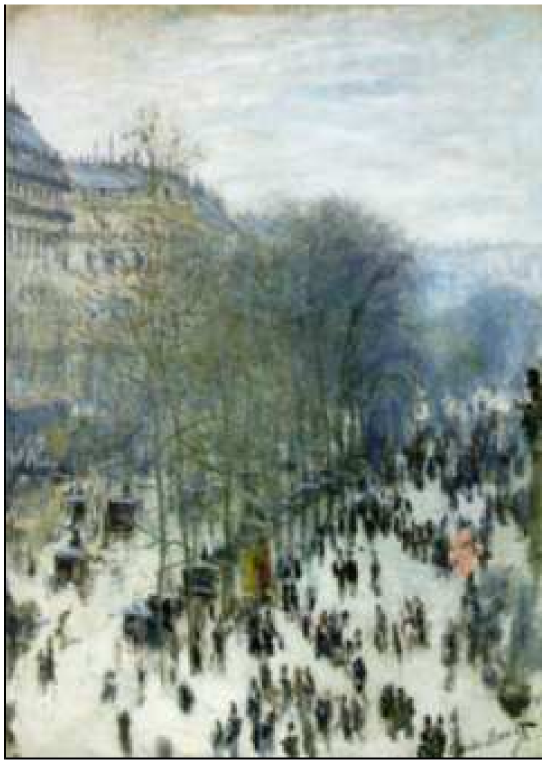
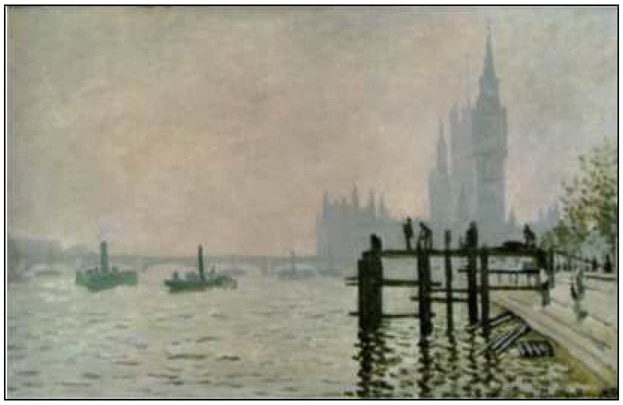


Fig. 19. The shading process.



(a) Claude Monet's Boulevard of Capucines.



(b) Claude Monet's Thames by the Wesminster.

Fig. 20. Paintings display the atmospheric quality and depth achieved by reducing stroke and color detail for distant objects [8].

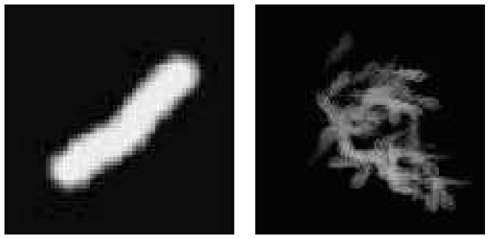


Fig. 21. Stroke sample images. These are used as alpha masks on the actual color map. The alpha values range from 0 to 1, where black is equal to an alpha value of 0 and white is equal to 1.

CHAPTER IV

IMPLEMENTATION

The painterly look will be achieved by applying a surface shader to the objects in the MAYA scene. The shader will be scripted in Renderman shading language. A surface shader in Renderman returns a final color and opacity calculated from user defined functions.

Renderman provides a powerful and flexible shader scripting environment. Created primarily with the intention to generate photorealistic images in computer graphics, its flexibility allows one to achieve non photorealistic results as well. The Renderman software is the industry standard for use in rendering and shading CG objects.

Maya will be used as the main scene building program. All the modeling and lighting needed to create the scene will be done within or imported into Maya. Maya offers comprehensive modeling lighting and animation tools. The Renderman shading pipeline can be seamlessly integrated within Maya through Renderman's MTOR (Maya to Renderman) interface.

All the examples in this thesis have been produced on *SGI-Iris, Linux* and *Windows* platforms. Maya scene files in .mb or .ma formats may be used as inputs for shading. The MTOR plugin must be installed in Maya for the using the Renderman interface within Maya.

IV.1. Shader Implementation

The Renderman shading language is used to implement the surface shader. We adopt the layered approach to build the shader. In this approach the final surface pattern needs to be broken down into layers of simpler patterns. Thus, instead of writing a shader directly for the final more complex pattern, each simpler pattern can be written as a layer. Layers are defined from back to front and successive layers are composited over each other. For the painterly shader the underpainting, outline and an user defined numbers of brush stroke layers are all computed individually. Each layer computes a layer color and layer transparency that is used for compositing.

Underpainting- The underpainting is the second layer of the shader. Although the primary purpose of the underpainting is to establish the form of the objects in the scene, the program uses the underpainting to give objects an irregular hand drawn look.

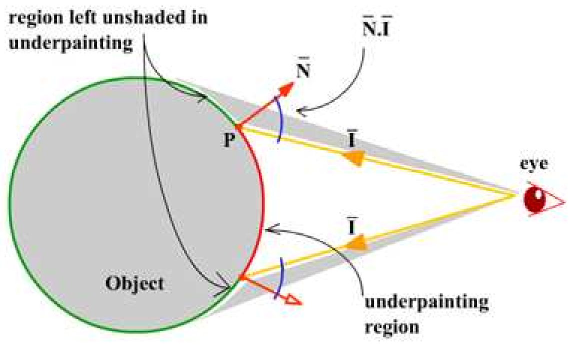


Fig. 22. Threshold for underpainting.

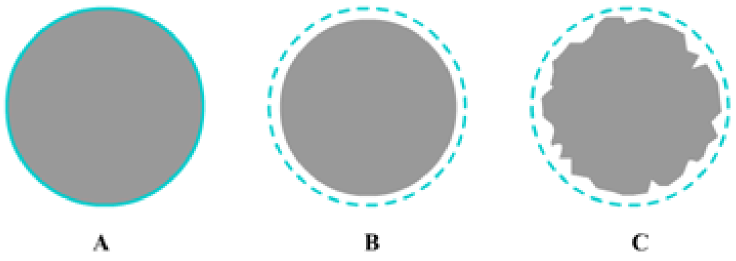


Fig. 23. Underpainting creates the illusion of discontinuous border.

One of the most important goals in creating the painterly shader was to see that the objects in the scene did not have very sharply defined boundaries. This is hard to achieve in a surface shader as there is no way to shade a point that lies outside the surface of the object. Previous solutions have involved the use of particles to make the boundaries discontinuous. The solution presented here is to create the illusion of broken edges by keeping the under-painting within a threshold calculated by the angle between the view vector and the surface normal of the point being shaded. As shown in Figure 22 the underpainting is done only in the region where the angle between the viewvector (I) and surface normal at the point being shaded (N) is a value less than 90. Figure 22A shows an object with underpainting threshold equal to 90, Figure 22B shows the object with an underpainting threshold less than 70, Figure 22C shows an underpainting threshold less than 70 + (a noise value). In Figure 23BC the actual boundary of the object is shown by the dashed line.

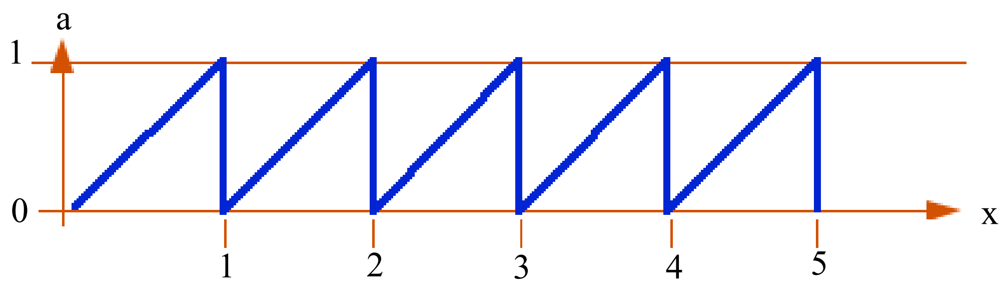


Fig. 24. mod(x,a), where a=1 [16].

Overpainting- The overpainting step involves applying the brush strokes over the underpainting layer. The user is required to provide the shader with three brush stroke images. These images are then tiled by the shader to create brush stroke layers. The mod function is useful in creating periodic patterns. mod produces a sawtooth pattern which repeats from 0 to a value based on the variable x as shown in Figure 24.

Figure 25B shows the use of mod to repeat a function described in Figure 25A in any given frequency value.

The repeat function is used to tile the brush image over value of frequency in the

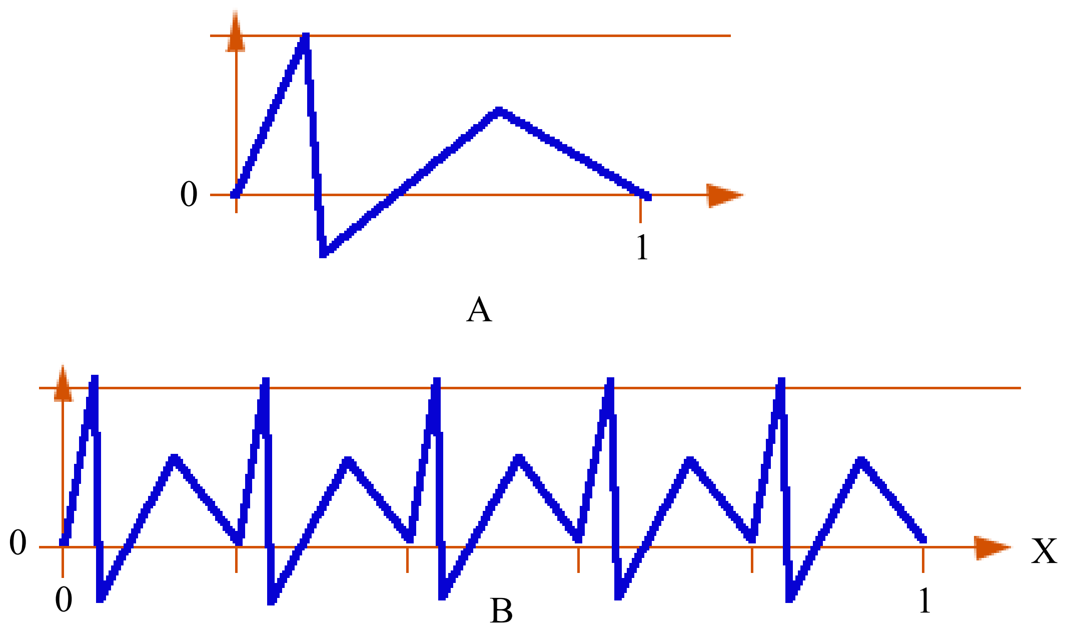


Fig. 25. Repeating a function using mod [16].

texture space. To repeat the tiles over the entire image mod is used.

Instead of using texture coordinates (s, t) which vary from 0 to 1 over the entire surface to compute a tile pattern, we use *tile texture coordinates (ss, tt)* which vary from 0 to 1 repeatedly over the surface. The *repeat* function is defined as:

repeat(x, frequency) = (mod((x) * (frequency), 1.0))

Example, ss=repeat(s,4) tt=repeat(t,4), the result is seen in Figure 26. The texture is tiled 16 times. Four times in the s direction and four times in the t direction. A noise function is added to the tiled ss tt values to give a randomness to the final resulting texture.

Layers and Compositing- Most surface shaders will consist of multiple layers. The accumlated color of all layers will be stored in a variable called surface-color.

Initially, a value is directly assigned to surface-color which will indicate the color of the background layer (layer 0). This value may be fixed by the shader-writer or definable by the user of the shader using an instance variable or the global Renderman Shading Lan-

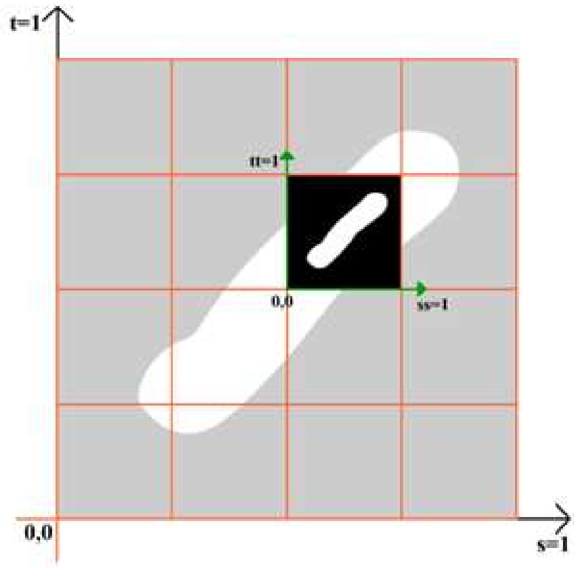


Fig. 26. Tiled texture coordinates ss tt within a single tile.

guage (RSL) variable Cs. Cs stands for the surface color in Renderman Shading language.

For each foreground layer, the variables layer-color and layer-opac represent color and opacity for the layer currently being defined (these variables will be reused for each layer).

```
/* layer n */
layer-color = function to calculate layer color
layer-opac = function to calculate layer opacity
surface-color = blend(surface-color, layer-color, layer-opac)
```

Each layer will be composited on top of the previous layers using the RmanNotes blend() function.

color blend(color a, color b, color x), where a is the surface color, b is the layer color and x is the layer opacity. This is illustrated in Figure 27.

The blend performs a component-wise linear interpolation between a and b based on the values in x. Note: blend() also supports the argument types supported by the RSL

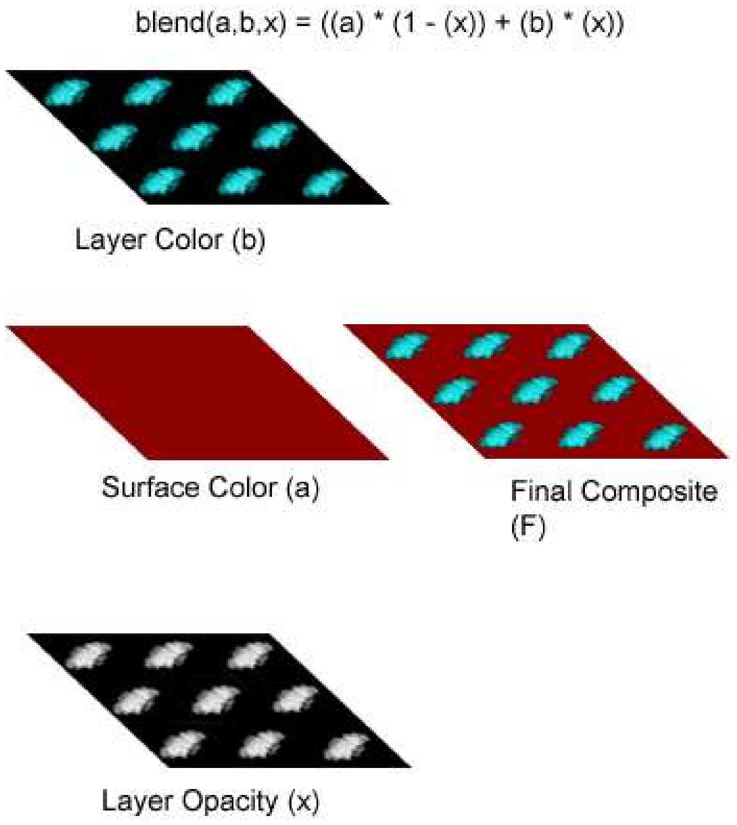


Fig. 27. Blending of layers.

function mix().

Figure 28 illustrates how the the blend function works.

The processes involved in the creation of a flat brush stroke texture starting with a single user defined brush image is summarized in Figure 29.

Texture file format- To access textures within a Renderman shader, texture files need to be in .tx format. This is the only texture file format supported by Renderman. Files in TIFF format may be converted to .tx using Pixar’s txmake utility.

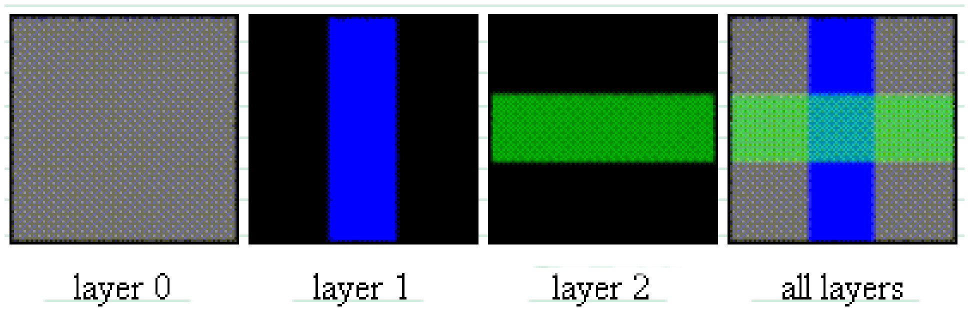


Fig. 28. Blending of layers [16].

IV.2. Illumination

As mentioned earlier, lights will be place in the scene in Maya. The painterly rendering program take the light position and angle created by the light ray and the point being shaded into account for coloring the brush strokes. The program uses *illuminance loops* to access the light information in the scene. Illuminance loop is a built-in Renderman function useful for making calculations involving lights in the MAYA scene. The loop runs through each light in a scene and makes user defined calculations using position, color, intensity, angle of incidence and other properties of the light source to affect the appearance of the surface shader. Figure 30 shows the relation between light angle and color calculations.

IV.3. Bump Mapping

Some traditional media artists paint with thick strokes of color. Therefore their paintings are not always flat entities. The thickness of paint adds to the texture created by the brush strokes.

This effect is created by the shader using bump mapping. A grey scale texture is created out of the final over painting layer and used as a bump map for the painting. Bumps

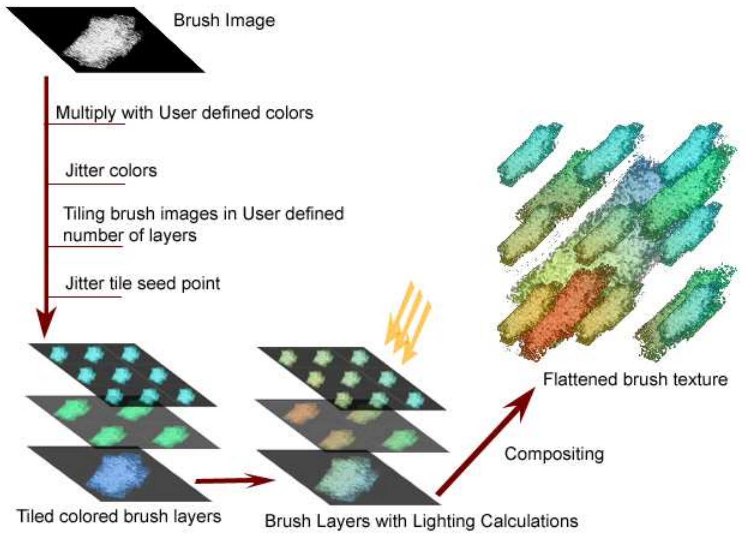

Brush Image
Multiply with User defined colors
Jitter colors
Tiling brush images in User defined number of layers
Jitter tile seed point
Tiled colored brush layers
Brush Layers with Lighting Calculations
Compositing
Flattened brush texture

Fig. 29. Creating the brush stroke texture.

are created by recalculating shading normals for the point being shaded. The new normal is displaced in the direction of the surface normal at the point being shaded by an amount dictated by the grey scale image. Lighting calculations are made with this new displaced normal. Figure 31 shows the visual differences between a scene rendered with and without bump map.

IV.4. Animating Shader Parameters

The shader itself will have modifiable parameters that the user can tweak to achieve the desirable look. Inputs for brush files, brush color, shadow color, diffuse color, outline thickness, brush rotation value are some of the parameters that would be user defined. These parameters can also be animated.

Animation- Many of the shader's parameters can be animated within Maya using a simple connection expression or TCL command in Renderman. The process for making

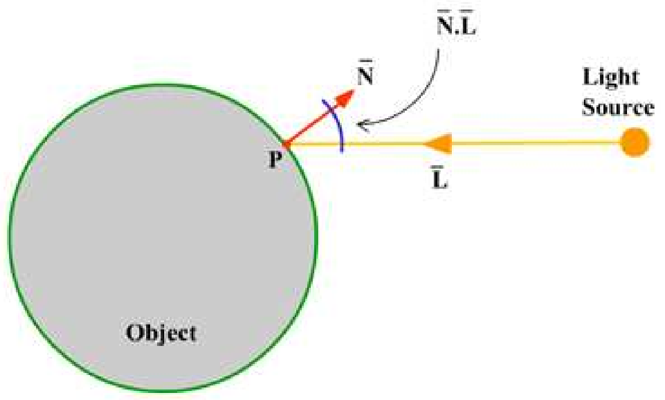


Fig. 30. Surface color based on the angle between incident light and normal at the point being shaded.

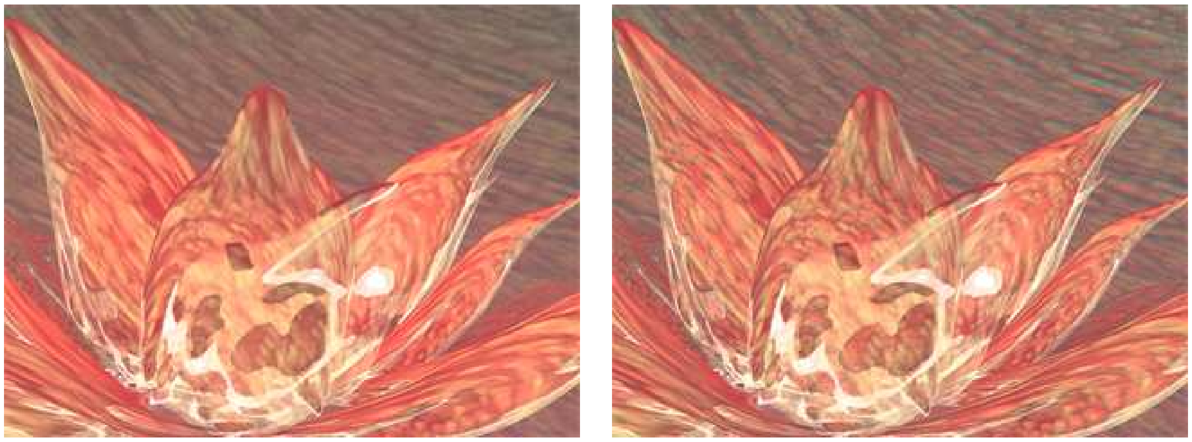
the Maya-Renderman connection involves the following steps.

1. Create a locator in the Maya scene.

2. Add attributes to the locator that may be modified via expression or keyframing to affect the Renderman parameter to be animated.

3. The general expression [”matter LocatorName.AtrributeName” AttributeType] added to the shader parameter TCL box will complete the connection from the Maya locator attribute to the Renderman shader parameter. For example adding the expression [”matter Locator1.brushrotation” F] to the desired parameter in the Renderman shader properties dialog box. This connects the user defined attribute ”brushrotation” belonging to Locator1 in Maya to, say, the brush rotation Value in Renderman. Thus, if the attribute ”brushrotation” is changed over an animated sequence the Renderman shader will update the rotation angle of the brush strokes in runtime.

   In the same way other attributes such as brush color, underpaintiong color, brush and underpainting opacities and noise may be animated.



(a) Without bump map (b) With bump map

Fig. 31. A scene rendered with and without bump map.

A second method for animating the brush strokes is by animating the UVs of the geometry. Since the shader places itself on the geometry depending on the position of the geometry UVs, therefore animating the UVs by rotating, translating or scaling will similarly affect the shader as well.

IV.4.1. User Interface

The standard Maya interface will be used for scene building. The shader developed for painterly rendering will be used within the MTOR work environment. Renderman's *slim palette* is an important component of the MTOR user interface. The painterly shader may be imported into the slim palette and applied to objects in the scene. The user can modify the shader parameters by changing the default values in the shader attribute editor. The shader attribute editor can be brought up by double clicking on the shader node in the slim palette. Figures 32-39 show screen shots of the Maya and Renderman user interface.

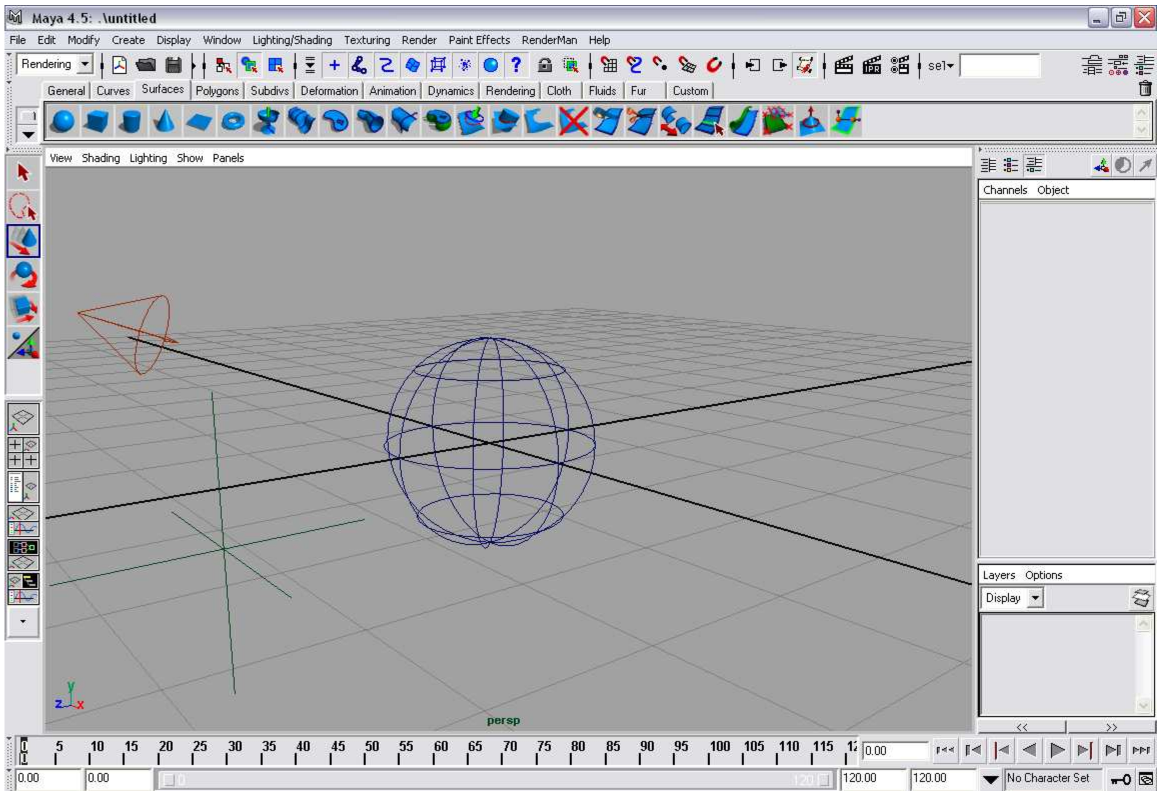

Maya 4.5: .\untitled
File Edit Modify Create Display Window Lighting/Shading Texturing Render Paint Effects RenderMan Help
Rendering
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering Cloth Fluids Fur Custom
View Shading Lighting Show Panels
Channels Object
Layers Options
Display
persp
No Character Set

Fig. 32. The MAYA interface.

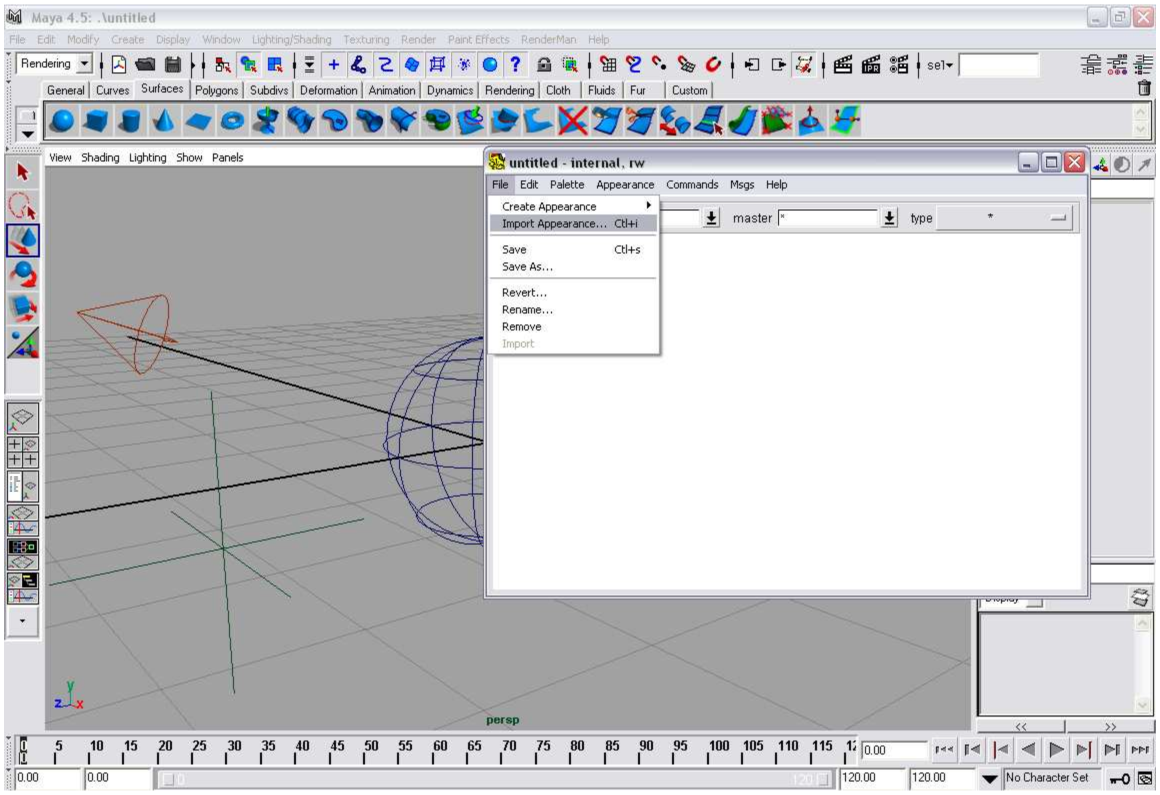

Maya 4.5: .\untitled
File Edit Modify Create Display Window Lighting/Shading Texturing Render Paint Effects RenderMan Help
Rendering
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering Cloth Fluids Fur Custom
View Shading Lighting Show Panels
untitled - internal, rw
File Edit Palette Appearance Commands Msgs Help
Create Appearance
Import Appearance... Ctl+i
Save Ctl+s
Save As...
Revert...
Rename...
Remove
Import
master
type
persp
No Character Set

Fig. 33. The Renderman slim palette loaded within the Maya interface.

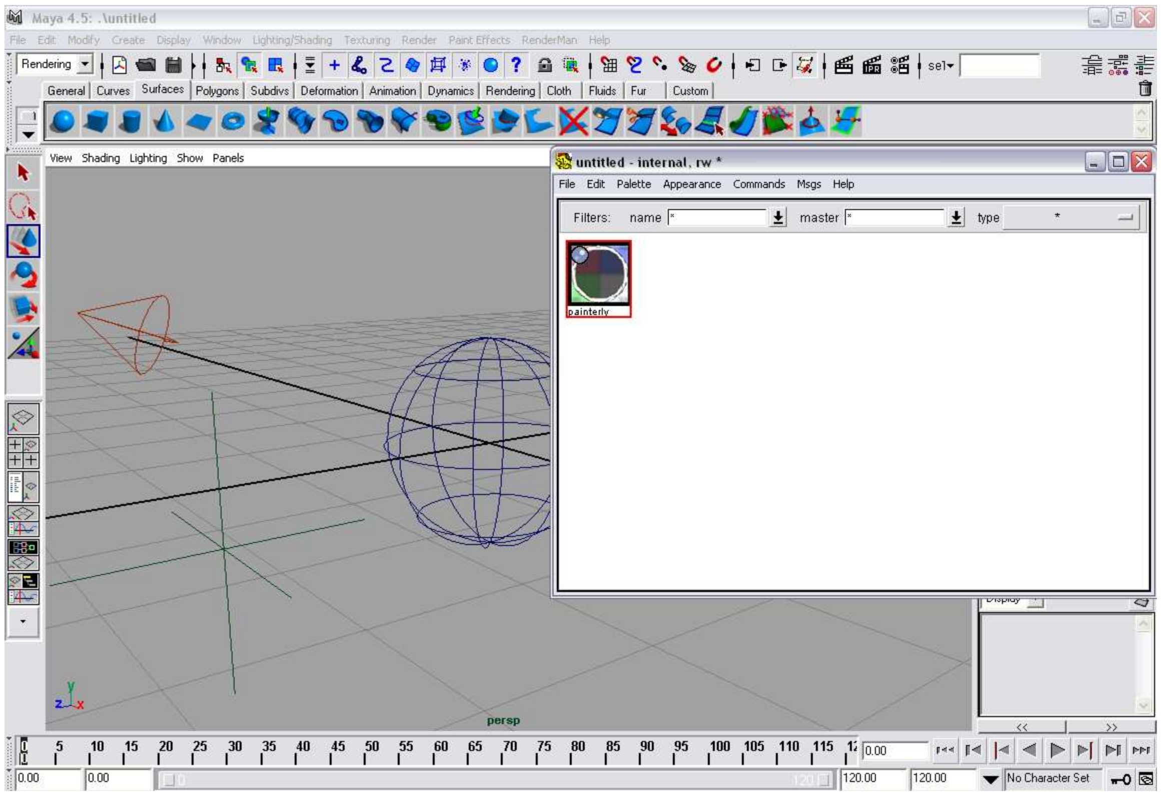
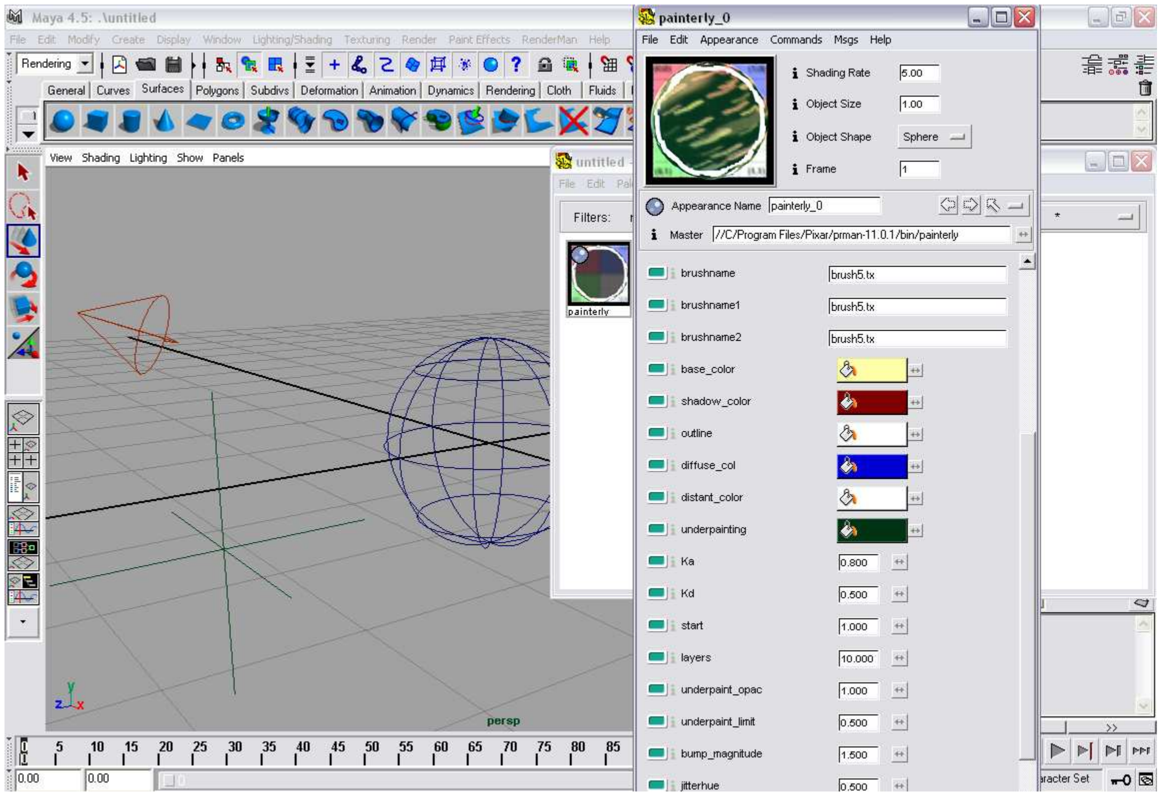
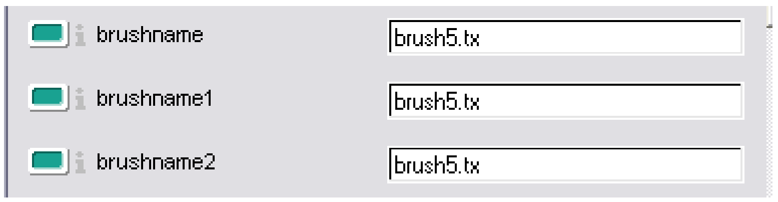

Maya 4.5: .\untitled
File Edit Modify Create Display Window Lighting/Shading Texturing Render Paint Effects RenderMan Help
Rendering
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering Cloth Fluids Fur Custom
View Shading Lighting Show Panels
untitled - internal, rw *
File Edit Palette Appearance Commands Msgs Help
Filters: name master type
painterly
persp
No Character Set

Fig. 34. The painterly shader is imported into the slim palette.


Maya 4.5: .\untitled
File Edit Modify Create Display Window Lighting/Shading Texturing Render Paint Effects RenderMan Help
Rendering
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering Cloth Fluids
View Shading Lighting Show Panels
persp
untitled
Filters:
painterly
painterly_0
File Edit Appearance Commands Msgs Help

Shading Rate	5.00
Object Size	1.00
Object Shape	Sphere
Frame	1
Appearance Name	painterly_0
Master	//C/Program Files/Pixar/prman-11.0.1/bin/painterly
brushname	brush5.tx
brushname1	brush5.tx
brushname2	brush5.tx
base_color	
shadow_color	
outline	
diffuse_col	
distant_color	
underpainting	
Ka	0.800
Kd	0.500
start	1.000
layers	10.000
underpaint_opac	1.000
underpaint_limit	0.500
bump_magnitude	1.500
jitterhue	0.500

Fig. 35. The painterly shader attribute editor may be invoked by double clicking on the shading node in the slim palette. The appearance of the painterly shader can be changed by changing the values of the shader attributes.



brushname	brush5.tx
brushname1	brush5.tx
brushname2	brush5.tx

Fig. 36. Three brush stroke images can be specified in the brushname, brushname1 and brushname2 channels.

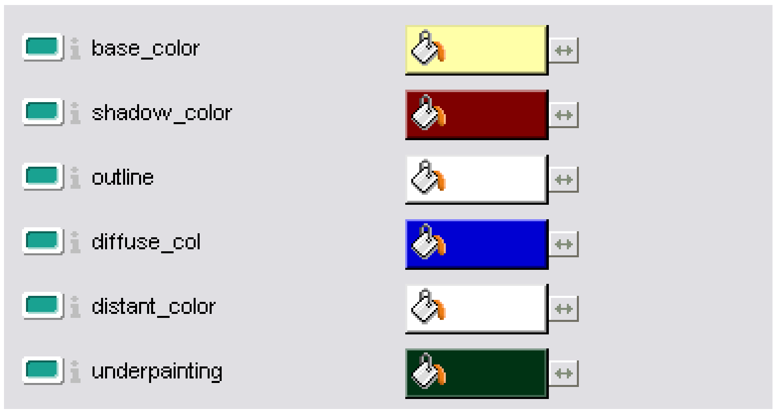


Fig. 37. Color controls for stroke, outline, underpainting, distance and diffuse color.

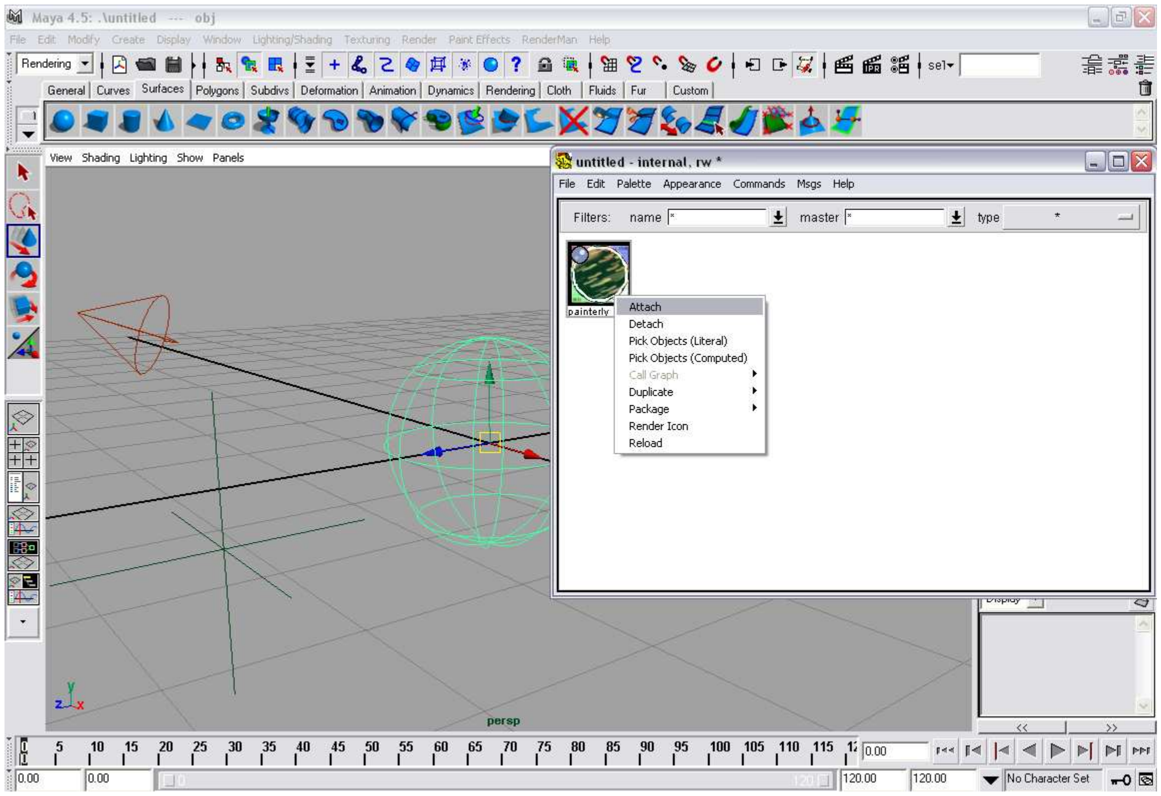

Maya 4.5: .\untitled --- obj
untitled - internal, rw *
File Edit Palette Appearance Commands Msgs Help
Filters: name master type
painterly
Attach
Detach
Pick Objects (Literal)
Pick Objects (Computed)
Call Graph
Duplicate
Package
Render Icon
Reload
persp

Fig. 38. Attaching the shader to the geometry by selecting the geometry and clicking the attach button on the shader right click menu.

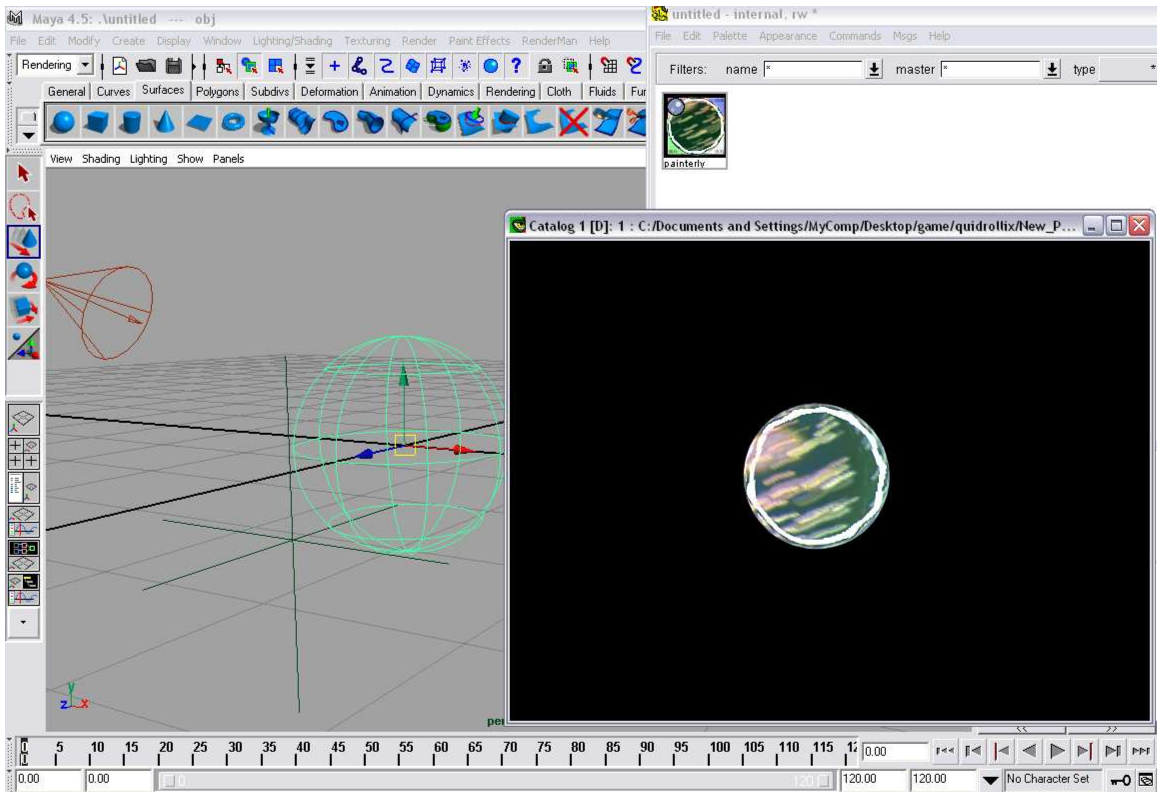
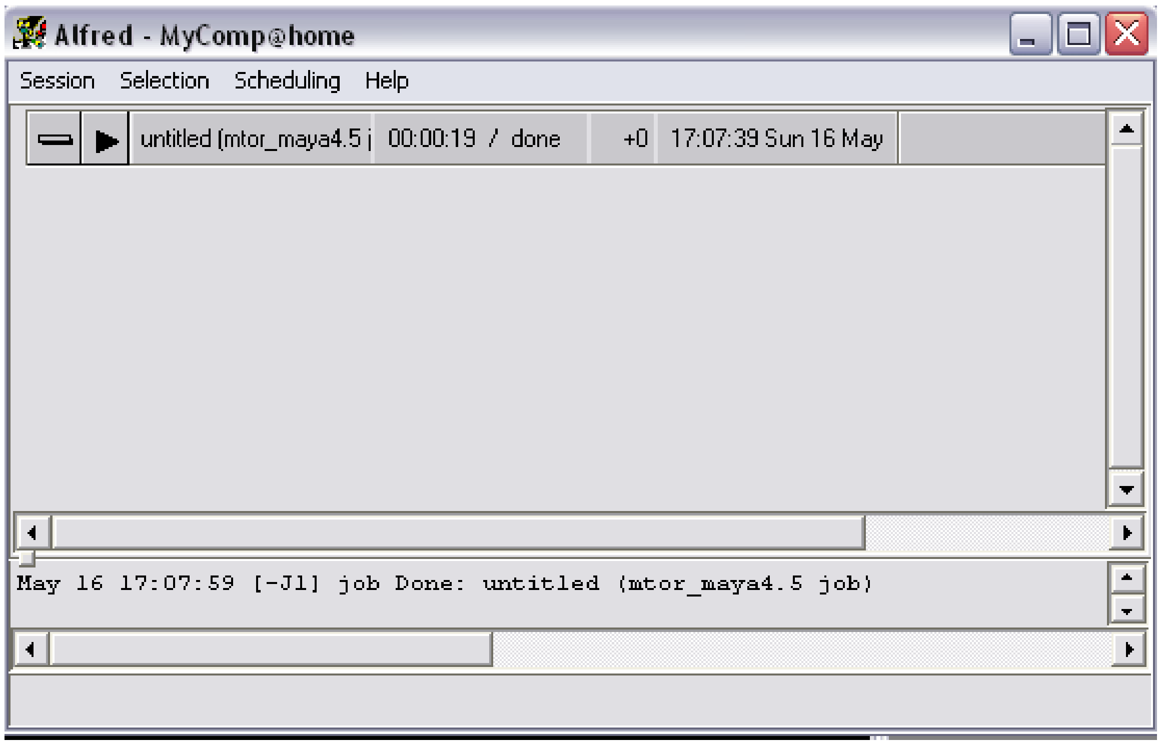

Maya 4.5: .\untitled --- obj
File Edit Modify Create Display Window Lighting/Shading Texturing Render Paint Effects RenderMan Help
Rendering
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering Cloth Fluids Fur
View Shading Lighting Show Panels
untitled - internal, rw *
File Edit Palette Appearance Commands Msgs Help
Filters: name master type
painterly
Catalog 1 [D]: 1 : C:/Documents and Settings/MyComp/Desktop/game/quidrollix/New_P...
No Character Set

Fig. 39. Clicking the *render* button on the *Renderman* menu tab starts the render. The rendered image is displayed on Renderman's *it* utility.



```
untitled (mtor_maya4.5 j   00:00:19  /  done   +0   17:07:39 Sun 16 May
```

```
May 16 17:07:59 [-J1] job Done: untitled (mtor_maya4.5 job)
```

Fig. 40. The render job status and progress can be managed on Renderman's *Alfred* utility.

CHAPTER V

RESULTS

As a proof of concept, all final images and animations were created in Maya and MTOR. All images and animations were rendered in a single render pass. The subjects chosen for the final renderings are two landscapes, a sculpture and a still life. These common subjects for conventional paintings and would thus be suitable subjects for testing the painterly shader.

The results demonstrate the use of different brush stroke images to achieve a variety of painterly styles. As is evident from the results shown, the brush strokes are the most distinct visual feature of the painterly shader. The strokes not only impart the images with textural richness but also give scope for stylistic variation to the images.

As mentioned earlier, traditional painters may choose to outline certain objects in their scene to emphasize them more strongly. The shader provides this utility by allowing the user to choose whether or not to outline the geometry being shaded. The user may also control the firmness of the outline that will be applied to an object.

The type of brush stroke images applied may also be used to create a visual likeness to several different media. Depending on the type of brush strokes used and the opacity of the strokes, effects similar to watercolor or dry media such as chalk or crayon may be created.

All the objects in each of the rendered scenes use different instances of the basic painterly shader. Different instances are created by changing brush stroke type, nature of outline, under painting color, over painting color and several other color parameters that are provided as shader attributes. The results attempt to demonstrate the use of the flexibility of the shader parameters for the creation of different styles of painterly work. In doing so, the shader establishes its ability to enable users to develop and express their ideas through individual styles.

Following are some significant results that have been achieved, by using the painterly shader presented in this thesis work:

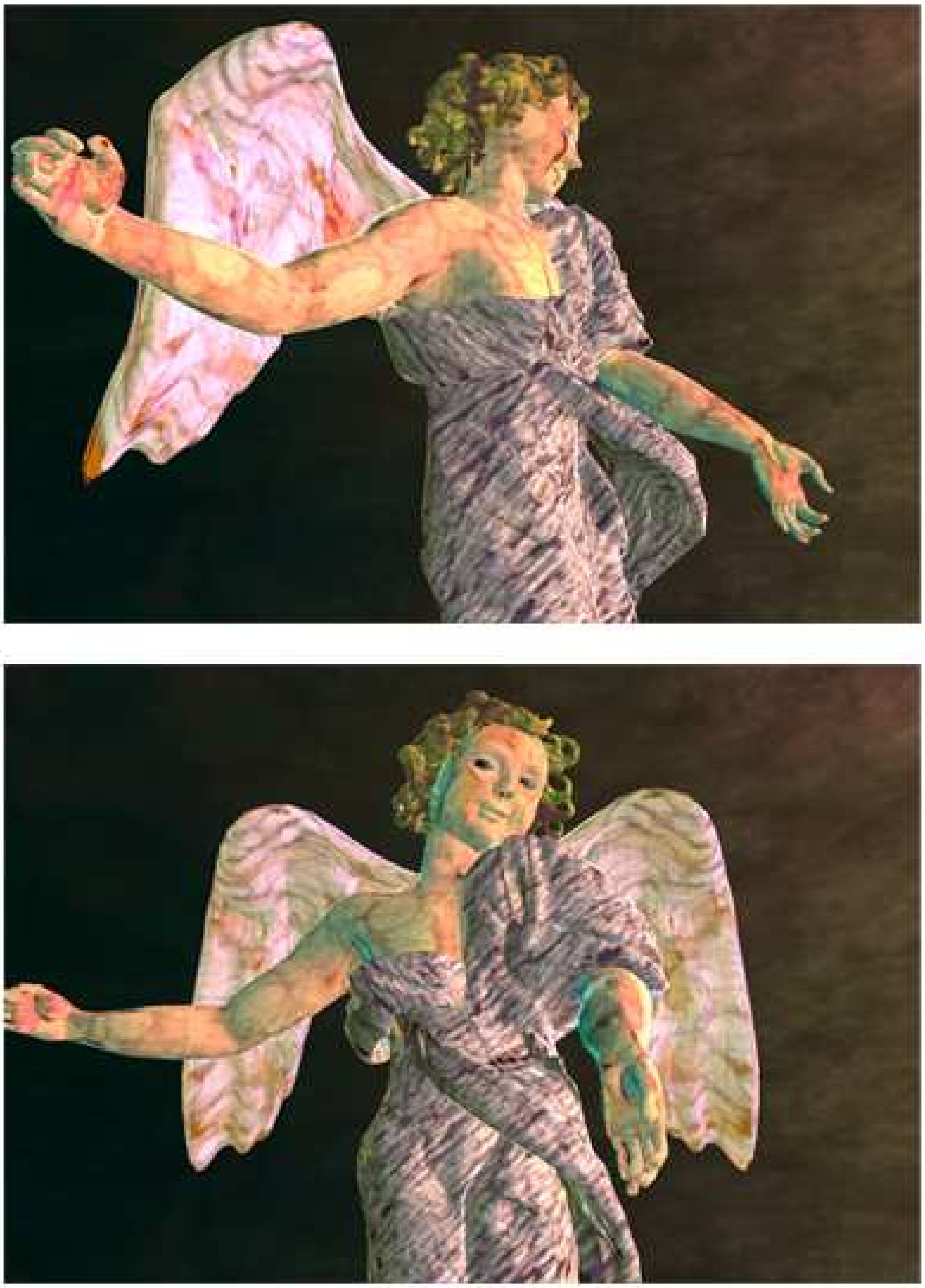


Fig. 41. Sculpture model using different brush strokes.

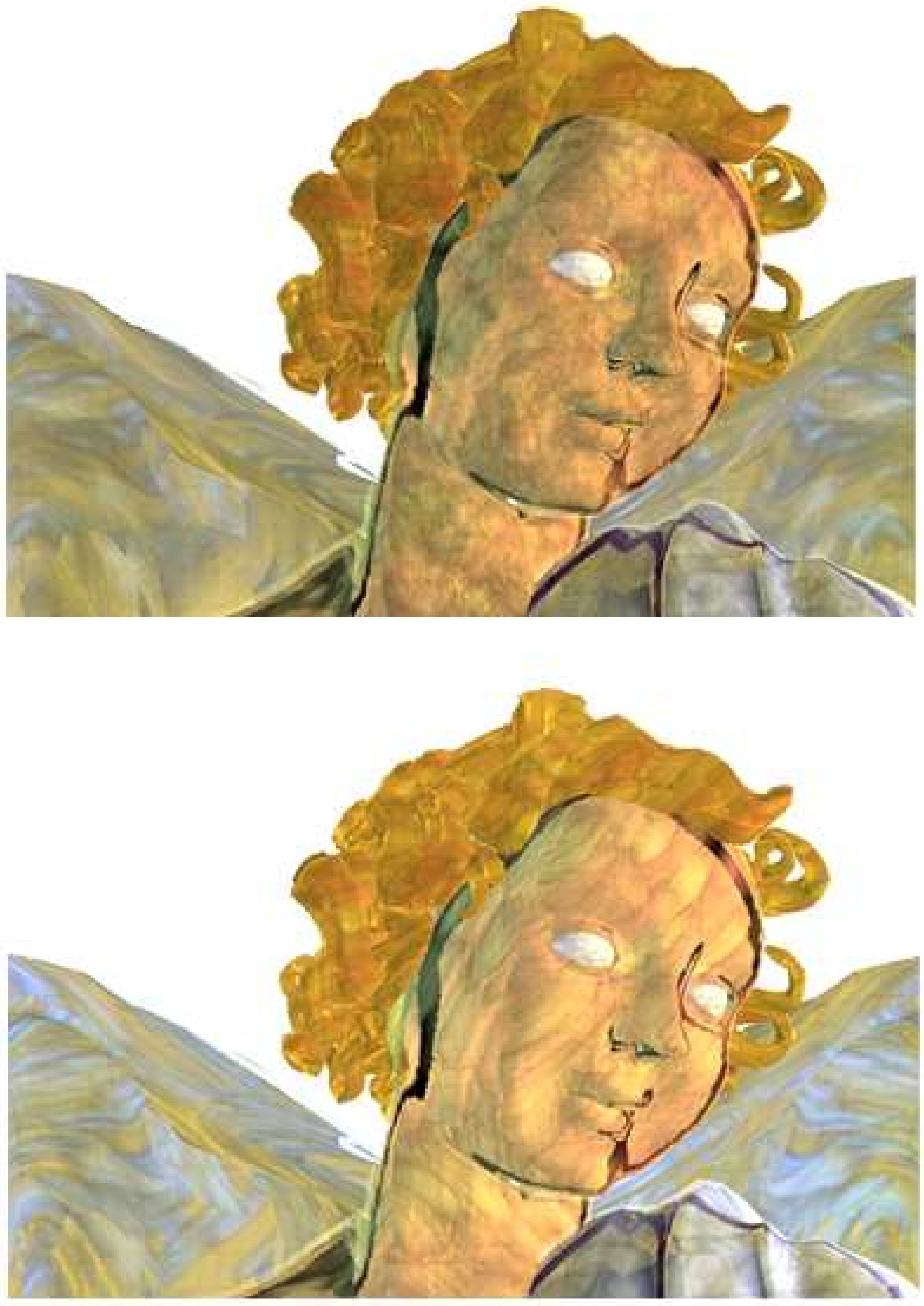


Fig. 42. Detail of sculpture model using different brush strokes.

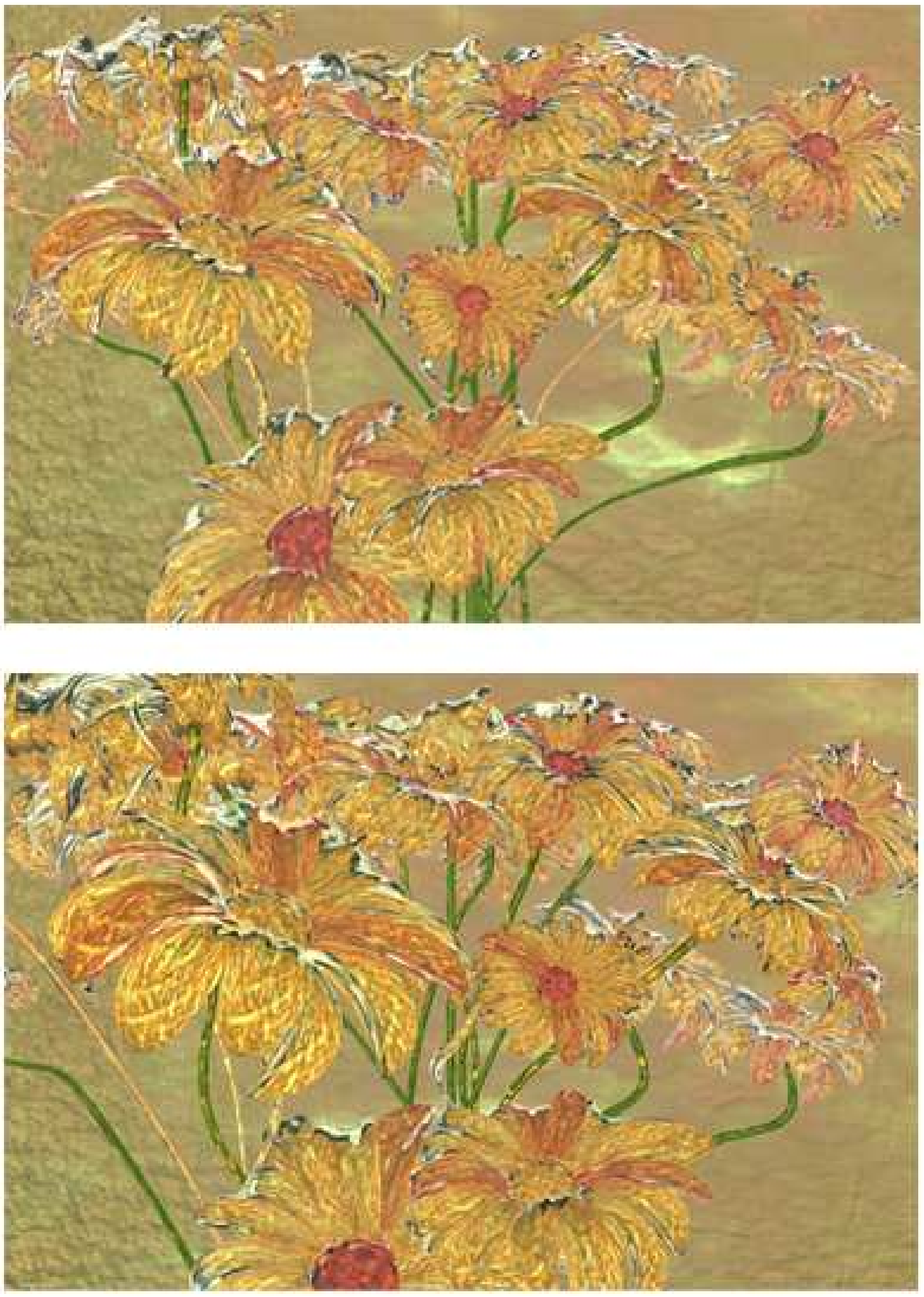


Fig. 43. Frames from animation of flower model.

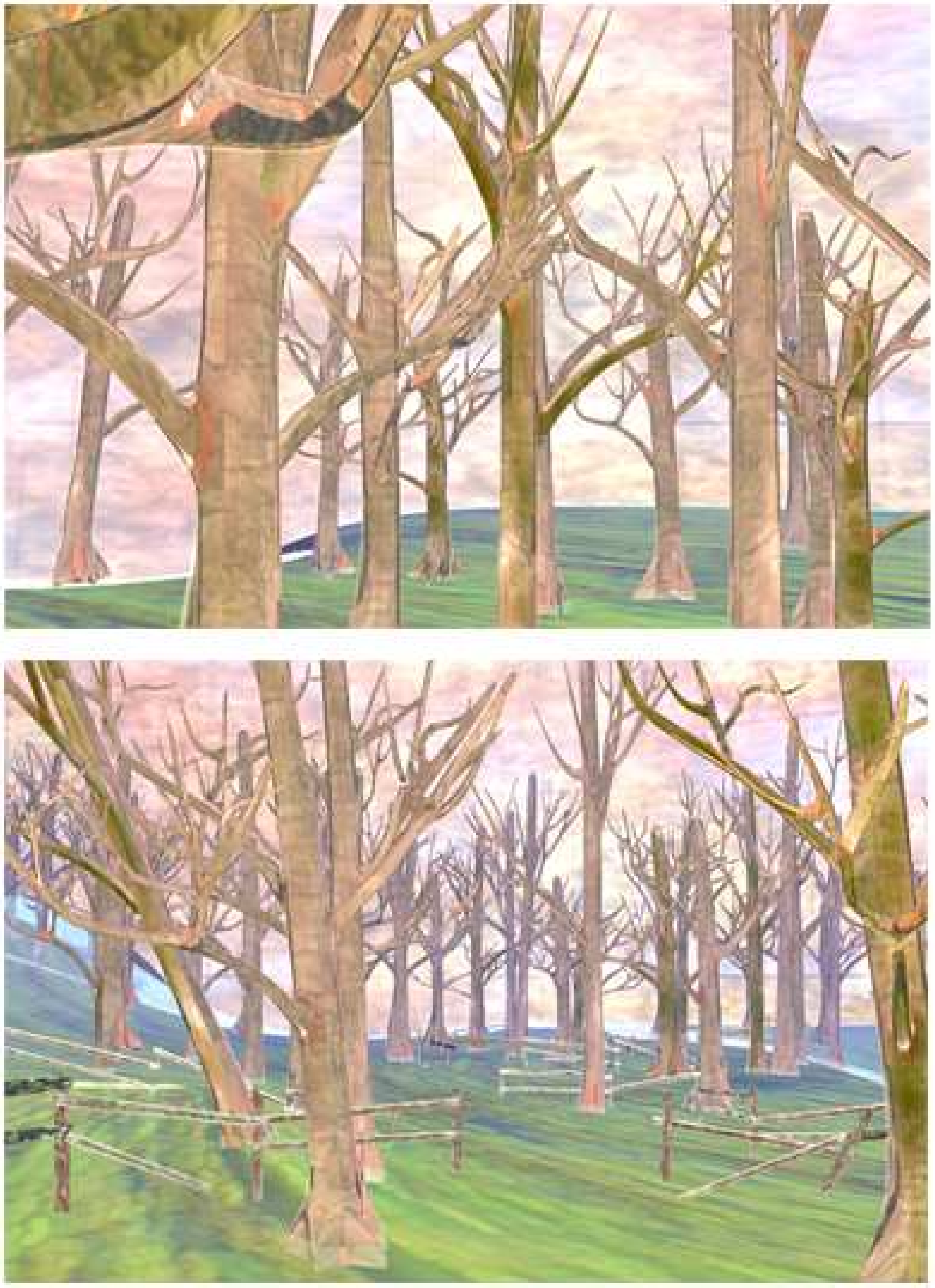


Fig. 44. Frames from animation of landscape model, showing visual depth.

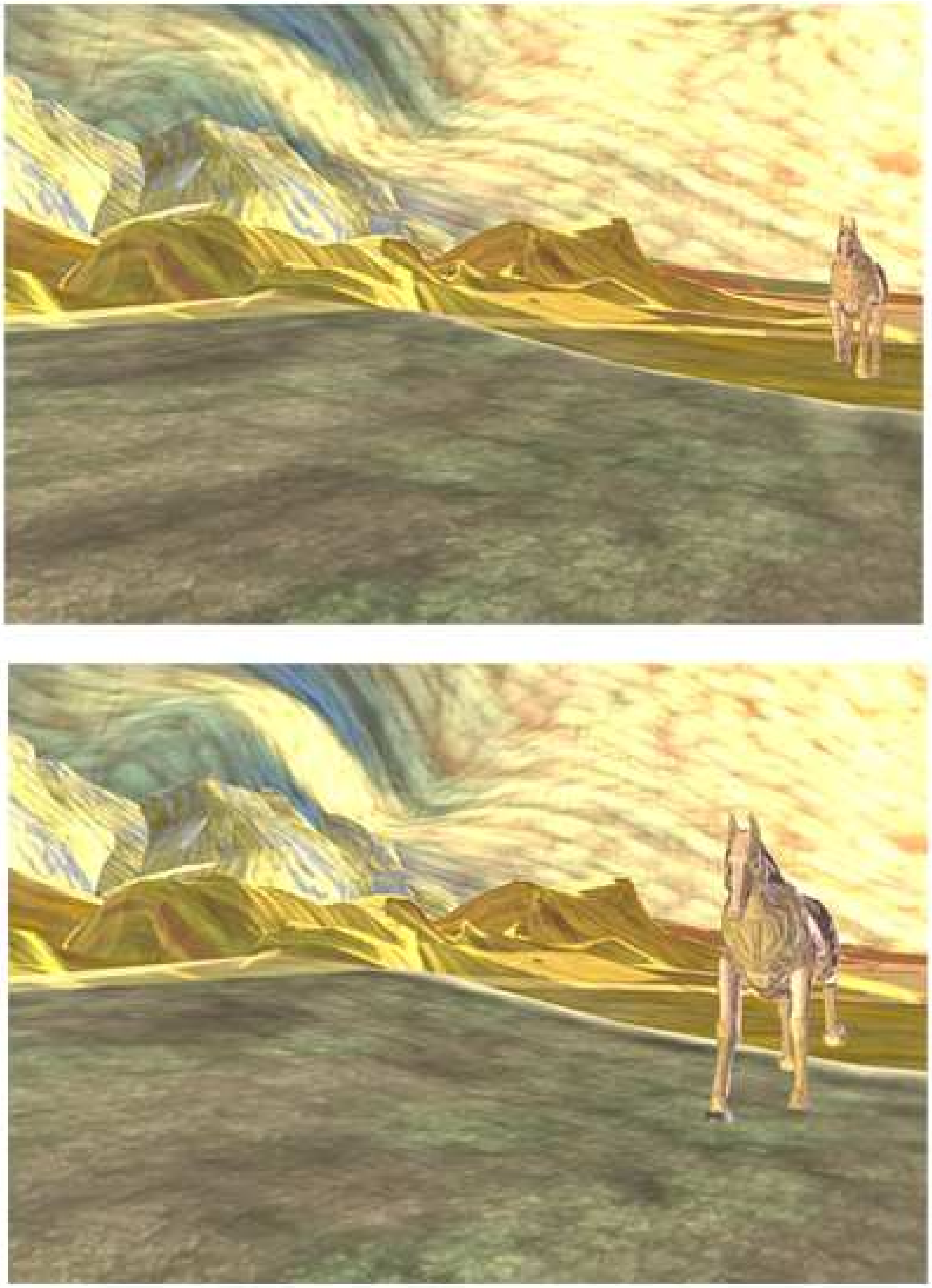


Fig. 45. Frames from animation of horse and landscape model.

CHAPTER VI

CONCLUSION AND FUTURE WORK

VI.1. Future Work

There are many directions in which possible future work can be carried on with respect to the work presented in this thesis.

A few suggested directions of work that would improve the visual quality of the final rendered images are:

- **Dynamic stroke depth-** The shader uses color to define depth. A more effective solution for depth would be to alter brush detail as well. Objects in the foreground would require to render in more detail and thus use more brush stroke layers than objects in the background.

- **Randomization of brush strokes-** Random repetitive application of brush strokes can be prone to pattern formation. This may be undesirable as distinct patterns will inhibit the painterly look. Improvements can be made to deal with better randomization and scaling of brush strokes. In this program the problem is handled by keeping the brushstrokes at random and low opacities so blending between successive layers would eliminate very distinct patterns.

- **Broken Edges-** Breaking up the edges more significantly-One of the limitations a surface shader has is that textures on it tend to give a ”gift wrapped” look. This shader handles this problem by leaving ”holes” or uncolored regions on the insides of the boundary of the object, thus visually creating discontinuous edges. Other ideas to get better broken edges would add to the painterly look significantly.

- **Brush Animation-** The brush animation in the painterly program can be done in expressions controlling brush rotation and translation. For closer shots in the scene this might sometimes be visually distracting. There may be some instances where it would just give a ”swimming textures” sort of feel. Future work can aim to make brush animation more sophisticated. A simulation based, fluid-like motion may add a lot to painterly animation.

VI.2. Conclusion

This thesis provides a new approach to painterly rendering. Previous work on painterly rendering used a particle based method or used post processing on rendered images to achieve painterly effects. This technique refers to concepts in previous work that uses 2D brush stroke images and uses them to form a shader. The shader based approach aims to simplify the painterly rendering pipeline and the MTOR interface makes usage very convenient. Keyable shader parameters make it easy to animate shader properties. The rendered images will demonstrate the properties of painterly work such as broken silhouettes, brush stroke textures and sizes that define surface properties and abstraction of composition.

The main painterly aspect that the shader attempts to capture is the use of discreet brush strokes to create a scene. The results of this thesis may therefore be likened to Impressionist paintings with respect to brushwork. Artists use various techniques to create an individual style of expression in painting. In the painterly shader, apart from color, brush strokes are significant as they give the artist some extra freedom in expressing individuality in style. Visually brush strokes serve to create textural richness in the renders.

One of the primary objectives of the shader is to automate the process of creating painterly renders directly from a 3D scene. Automation, while making usage very easy, comes at the cost of loosing some artistic control. The artist cannot be as willful or enjoy the same interactivity as conventional painting. Therefore, controlling characteristics of the render, like color and brush strokes at a micro level can, is not easy. The user can, however, make these artistic decisions with respect to the overall look of the scene by modifying the shader parameters. This is an important issue to understand for effective usage of the shader and for creating images that can be beautiful, expressive and very painterly.

REFERENCES

[1] Algorithmic Arts, ”Algorithmic Arts: Expressive Effects”. http://www.hollywoodjesus.com/fantasia2000.htm, accessed: April 2004.

[2] D. Bruce, ”Fantasia 2000”. http://www.hollywoodjesus.com/fantasia2000.htm, accessed: April 2004.

[3] C. J. Curtis, ”Loose and sketchy animation”,*Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pp.145, 1998.

[4] Gnomon Workshop, Inc, ”The Gnomon Workshop”. http://www.thegnomonworkshop.com/tutorials/transparency-shadows/trans-shadow.html/, accessed: September 2003.

[5] A. Gooch, B. Gooch, P. Shirley and E. Cohen, ”A non-photorealistic lighting model for automatic technical illustration”, *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, p.435-446, July 1998.

[6] J. Janson, ”Underpainting”. www.jasonmichaels.com/ mixed/mixed1.html, accessed: April 2004.

[7] P. Litwinowicz ”Processing Images and Video for an Impressionist Effect”, *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 407-414, August 1994.

[8] H. Mark, ”Welcome to Mark Harden’s Artchive”. http://www.artchive.com/welcome.htm, accessed: March 2004.

[9] J. Michael, ”Jason Michael’s Photography + Design”. www.jasonmichaels.com/mixed/mixed1.html, accessed September 2003.

[10] B. Meir, ”Painterly Rendering for Animation”, *International Conference on Computer Graphics and Interactive Techniques*, vol. 33, no. 2, pp. 477-484, 1996.

[11] B. Meir, ”Computer Artists Who Work Alone”, *IACM SIGGRAPH Computer Graphics*, vol. 33, no. 2, February 1999.

[12] Y. Minako, ”Non Photorealistic Shading in Maya”, July 2002. http://www.noboundrees.com/yinako/tut/nprs/nprstut.html, accessed: March 2004.

[13] S. Monesi, ”What Dreams May Come”. whttp://www.monesi.com/sergio/movies/nov98/whatdreams.html, accessed: April 2004.

[14] E. Pendleton, ”Still Life Oil Painting Lessons for Students”. http://www.elinart.com/fact/still.html, accessed: April 2004.

[15] S. Remark, ”Waking Life”. http://www.hollywoodjesus.com/wakinglife.htm, accessed: April 2004.

[16] RManNotes, ”RMannotes”, http://accad.osu.edu/ smay/RManNotes/rmannotes.html, accessed: February 2003.

[17] G. Winkenbach and D. H. Salesin, ”Computer Generated Watercolor”, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, p.91-100, 1994.

[18] G. Winkenbach and D. H. Salesin, ”Computer Generated Pen and Ink Illustrations”, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pp.91-100, 1994.

VITA

Kaushik Pal

3528, W 32 Street

Apt- 306

Minneapolis, MN 55416

Education

M.S. in visualization sciences, Texas A&M University, College Station, August 2004

B.Arch, Birla Institute of Technology, Mesra, May 2000