The Simulation of Art in Computer Graphics

Gabriel Wurzer
mccool@cg.tuwien.ac.at

Institute of Computer Graphics
Vienna University of Technology
Vienna/Austria

Abstract

If rendering is the process of going from an object description to an image, art can be considered to be the act of transforming a mental image into a real one, using the different techniques and tools. The simulation of this process, generally referred to as *non-photorealistic rendering,* is subject to this report.

Keywords: art, non-photorealistic rendering, pen-and-ink, painterly rendering, cartoon rendering

1 Introduction

In art, the most basic element is the **point**. It can have multiple appearances, such as a *sharp-cornered spot*, a *blurred spot*, or a *stain* [Koschatzky '90] (see Figure 1). The **line**, as next-higher element of drawing, *out-lines* (hence the name!) the shape of an object (this is often referred to as *contour*). It follows its boundaries, either as a closed line or in the form of multiple *line segments* (see Figure 2). Each object has distinguishing marks which are used by our vision system to recombine a pair of lines into mental objects. A form surrounded by lines is seen as an **area** (compare with Figure 2). The simple heuristic the brain uses for that is to take the *simplest form* for a given set of lines. This means that if details are left away, the form gets *clearer* and *better recognizable* (this is quite the opposite of what most people think).

Artists use various **drawing tools** to influence the appearance of their artwork: Pens, pencils, chalks, brushes or even fingers give different shapes to the produced strokes. Furthermore, the different types of the *pigments* used (e.g. ink, graphite, watercolor) determine their color and opacity.

- Pens and pencils are widely used for sketches and technical illustrations. Due to the thin nib, sharp lines can be produced. The main pigments are sepia, traditional ink and Indian ink (pen) as well as graphite (pencil). Contrary to pens, various tones can be produces with pencils depending on their *hardness.*
- Chalks and charcoals are used in portrait paintings and sketches. As for pencils, different tones are available (according to the pigment mixture).
- Brushes come in multiple shapes and are made of various materials (e.g. horse hair, marten hair). The pigments are available separately, using water or oil as binder. The application of brushes is referred to as *painting* (as opposed to *drawing*, which uses non-fluid pigments).

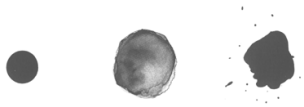


Figure 1: Types of points

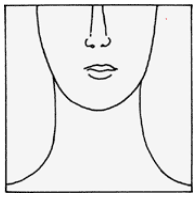


Figure 2: A set of lines, which we see as 'face'

The placement of the strokes is called **drawing technique** (*e.g. Hatching*, *Sumi*, *Pointilism*). Apart from this technical side of drawing, **selection, composition** and **abstraction** of a scene adds *meaning* to a piece of artwork, and is therefore considered to be the most important subject for an artist.

The simulation of art in computer graphics, often referred to as *non-photorealistic rendering,* has originally concentrated on the creation of **Pen-and-Ink Style Drawings** for CAD and illustrations. With growing interest in the field, researchers have also tried to mimic the style in which famous artists produce their artwork ("**Painterly Rendering**") not only for still images, but also for *animations.* This has lead to a strong influence of the movie industry on the field, mainly in the form of **Cartoon Style Renderings**. In this report, an overview of these techniques is given.

Overview

The paper is organized as follows: Section (2) deals with the generation of *Pen-and-Ink Style Illustrations.* In section (3), approaches producing *Painterly Renderings* are presented. *Cartoon Syle Renderings* are dealt with in section (5). In order to sum up, a categorization of non-photorealistic techniques is given in section (6).

2 Pen-and-Ink Illustrations

2.1 Hatching and Real-Time Hatching

In art, hatched illustrations are often used for studies and sketches. Their beauty lies in the simplicity with which they can convey both material, tone and form. Consequently, to produce these kinds of renderings has always been a vital goal to the computer graphics community.

2.1.1 The Beginnings

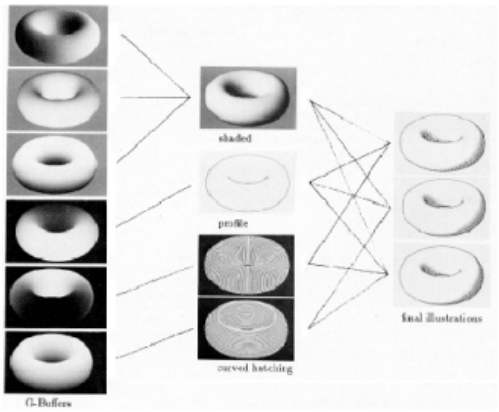


Figure 3: G-Buffers (left) in conjunction with unidirectionally hatched objects (middle) are used to produce hatched illustrations (right)

One of the first approaches that tried to render images interactively in line-art style was on "**Comprehensible rendering of 3D-Shapes**" [Saito, Takahashi '90]: By applying 2D image processing operations (e.g. *edge enhancement*) to a set of projected images (the so-called geometric buffers or '**G-Buffers**'), contour lines, edges and hatches can be visualized (see Figure 3). A similar approach which uses *half-toning* as such an image processing operation was presented in [Streit, Buchanan '98].

It is crucial to mention that this approach did not try to copy the way in which artists work, but concentrated on the operations necessary to produce a more illustrative look, for example in CAD programs. Similar approaches that change the shading model for that purpose were presented in [Gooch et al. '99, Gooch et al. '98] (see Figure 25). *Information flow* in such a sketchy modeler is discussed in [Strothotte et al. '94] and [Schumann et al. '96], *enhancement* of ordinary renderings with line-art illustrations in [Dooley, Cohen '90].

"**Computer-Generated Pen-and-Ink Illustration**" [Georges Winkenbach '94] was the first paper which investigates how pen-and-ink illustrations are made by artists.

A *Stroke* is produced by moving a nib with varying pressure along a slightly jagged path. Multiple strokes form a *stroke texture* (see Figure 4), and multiple stroke textures are painted over eachother to produce a desired tone. If the scale of the texture is increased, more strokes are necessary to produce the same tone.

Figure 4: Strokes indicate tone as well as texture. The more strokes are blended, the darker the tone is.

Outlines are special types of strokes which give a hint of the size and shape of an object and emphasize areas of interest (see Figure 5): *Boundary outlines* surround an object, while *Interior outlines* are used to express shadow direction and give view-dependent accents to the illustration.

Figure 5: A house rendered with outlines

2.1.2 Further Development

The key for real-time non-photo-realistic computer graphics often lies in finding its **silhouette** (see Figure 6). Mathematically, a *silhouette edge* is defined as edge connecting front-facing and back-facing polygons. An approach that uses a fast probabilistic identification of silhouette edges in object space in order to render line-

art in real time was presented by Markosian et al. [Markosian et al. '97].

Interframe coherence of the silhouette edges and a fast visibility determination based on Appel's hidden-line algorithm [Appel '67] is used to speed up the drawing process. The algorithm identifies large (and therefore more significant) silhouettes with higher probability than smaller ones. Only a small fraction of silhouette edges needs to be examined, which further adds to the speed. A method for rendering silhouettes purely in screen space is discussed in [Michael Cohen '99]. A hybrid approach that uses both object space and screen space was introduced by Northrup et al. [Northrup, Markosian '00]. It first finds the silhouette in object space and then continues to analyze the projection of these silhouette edges, producing smooth stroke paths in screen space which are then rendered.

Figure 6: A probabilistically rendered silhouette (using the approach by the Markosian et al.)

Winkenbach and Salesin [Winkenbach, Salesin '96] propose a pen-and-ink system specificly for **parametric surfaces** (see Figure 7).

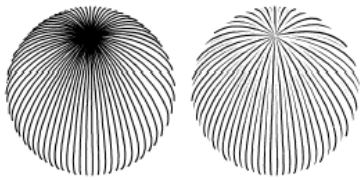


Figure 7: Parametric surface rendered normally (left) and with constant-density hatching (right)

The central problem the approach addresses is how to maintain constant tone over a surface which is bent, thus leading to the accumulation of more strokes (Fig. 7, left). The solution lies in what the authors call *constant-density hatching*, a technique with which the stroke thickness is adjusted to give the same tone over the surface (Fig. 7, right).

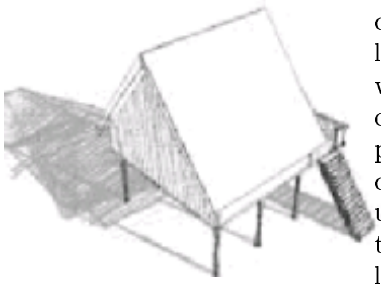


Figure 8: Pencil Illustration

The physical simulation of **graphite pencil drawing** is subject to the work by Sousa and Buchanan [Sousa, Buchanan '99]. They propose a rendering framework that encapsulates three levels of drawing: First, there is the simulation of the drawing materials (*low level*): The distribution of lead particles over a piece of paper is computed, with multiple layers being rendered above each other. Furthermore, different pencil hardnesses, pencil tip shapes as well as the structure of the drawing paper is taken into account (refer to Figure 8). At *medium level,* the simulation deals with the placement of the strokes and drawing of outlines. Composition of the scene and rendering as a whole is dealt with at *high level.*

Line direction is a critical factor in hatching. Girshik et al. [Girshick et al. '00] have provided evidence that line direction affects surface perception (e.g. curvature of a surface). Salisbury et al. [Salisbury et al. '97] have therefore proposed a system for creating pen-and-ink style renderings with orientatable strokes. By aligning the direction field with the surface orientation, a more expressive appearance of pen-and-ink illustrations can be achieved (see Figure 9).

Praun, Hoppe et al. [Praun et al. '01] present an approach which manages to produce **hatched illustrations in real-time**. Producing an animation out of a set of individually hatched images poses the following problems:

Figure 9: A stack of books rendered with various line directions

- There is only limited time for the computation of each rendered image
- The strokes need to be *frame-to-frame coherent*, or else the animation will have a flickery look
- The strokes need to maintain the desired tone even if the object moves toward or away from the camera. This is done by varying the density and width of the desired strokes.

The approach pre-renders hatch strokes into a sequence of mip-mapped images corresponding to different tones (**Tonal Art Maps**, TAMs), which are then used to texture the surfaces (see Figure 10).

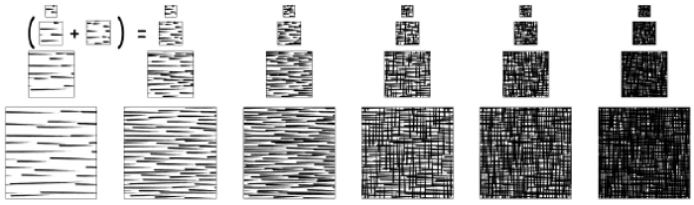


Figure 10: Tonal Art Maps: Strokes in lighter images are subsets of those in darker ones.

A triangle is drawn by blending several TAM images over each other using *multi-texturing* (see Figure 11). Each texture image is weighted using the lighting computed at the vertices. Thus, tones can be varied over each triangle. The use of Real-Time hatched illustrations for **3D gaming** has been researched by Freudenberg et al. Their approach uses a per-object specification of rendering style in order to optimize performance. Previous work on the subject [Alex Mohr '01] focused on intercepting OpenGL calls in order to replace them with sketched-style rendering calls. However, frame-to-frame coherence could not be reached with this method, resulting in a fuzzy and disturbing look. The approach leaves the specification of discontinuities on the 3D mesh to the artist, and introduces various simplifications in determining the visibility of the silhouette edges. Surface detail is added via mip-mapping (*Hatch Maps* and *Ink Maps*), with smaller images containing fewer lines in order to achieve constant tonality. Conventional shading is used for colorization of the objects.

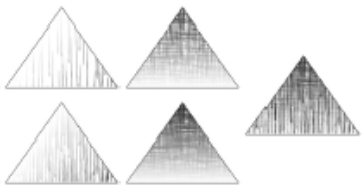


Figure 11: Blending TAM images on a triangle

2.2 Other Forms of Pen-and-Ink Illustration

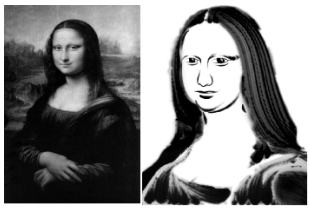
Apart from hatching, numerous other approaches are used to generate pen-and-ink style illustrations. Der-Loi Way et al. [Way et al. '01] give a way to generate **Chinese landscape and portrait paintings** semi-automatically.

The problem here is that Chinese painting tries to describe objects implicitly by abstracting them, which cannot be fully automatized. For landscape paintings, rocks are first manually outlined. Then, the interiors are automatically filled using two types of brush strokes (one for hard, rocky formations, another for smooth parts). For portrait paintings, a set of facial components is matched with the source image (see Figure 12).

Strassmann [Strassmann '86] has developed a method for drawing lines as **Sumi-e** style brush strokes by physically simulating the behavior of a wet brush on paper. A different approach for drawing painterly lines by deforming predefined brush-stroke images was presented in [S. C. Hsu '94].

Although technically not pen-and-ink, but often used for the same illustrative purposes, **charcoal drawings** have gained a respected reputation in art. Majumder et al. [Majumder, Gopi '01] generate such charcoal renderings by texture-mapping the mesh with grainy noise textures (see Figure 13). The contrast difference is exaggerated in order to emulate lighting effects and texture.

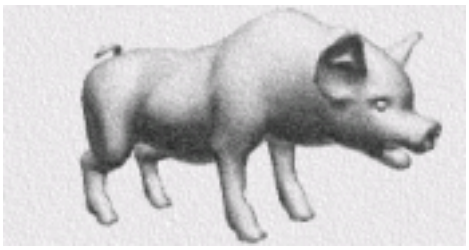
(a) Landscape



(b) Mona Lisa, Chinese style

Figure 12: Chinese painting

(a) Original art



(b) Charcoal rendering

Figure 13: Comparison of results.

3 Painterly Rendering

To paint in a style that matches that of great artists was long considered to be the domain of 2D image processing techniques. Recent approaches extend this idea to 3D, using particles to represent pigments on a surface.

3.1 Painterly Image Processing Techniques

Watercolor paintings are widely known for the diversity of applications and effects that can be produced with them (see Figure 14) . Since the simulation of all these drawing effects would be laborious, Curtis et al. [Curtis et al. '97] use a physical shallow-water model to simulate the distribution of pigments.

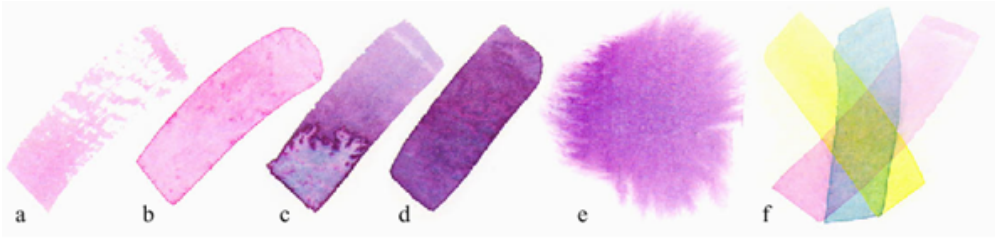


Figure 14: Watercolor effects - (a) dry-brush (b) edge-darkening (c) back-run (d) granulation (e) flow pattern (f) glazing

A system that produces **visible brush strokes** of various size to generate impressionistic painting style was presented by Hertzmann [Hertzmann. '98].

The author uses multiple passes over the image in order to give the image a more 'refined' look. Furthermore, strokes are rendered in random order so as to prevent an undesired appearance of regularity. From a source image, an artistic approximation is generated by progressively painting strokes of decreasing size over each other. The areas in which the strokes should be placed are derived from blurred versions of the source image. The resulting painting style vary depending on the user's specification (see Figure 15).

Figure 15: A painting produced by variable brush strokes and sizes

The application of the approach to **video** can be found in [Hertzmann, Perlin '00, Hertzmann '01] as well as [Litwinowicz '97]. In all of these papers, frame-to-frame coherence is an important subject. Because the brush strokes stick to the image plane rather than on the painted objects, a *shower door appearance* is the consequence (see Figure 16).

Figure 16: A *shower door* effect

There are dozens of other approaches that produce paintings in various styles. These techniques, however, are tailored specificly for that sole purpose. It would be nice if the computer could learn by itself how a specific painting style works, which would be **learning-by-example**. A recent paper that shows how this can be accomplished is the one on "**Image Analogies**" by Hertzmann et al. [Hertzmann et al. '01]. It uses *example images* to produce all kinds of image filters. The whole process of filtering can be broken up into two stages:

1. the design phase: the program is trained with both the filtered and the original version of an image. This is repeated with different images, and the resulting filter is stored.

2. the application phase: the filter is applied to a target image.

More precisely, the problem the approach tries to solve is: For a given unfiltered training image A, a filtered training image A' and an unfiltered target image B, compute the filtered target image B' (see Figure 17) .

Figure 17: The target image is filtered like the training images

Applications of this technique include

- traditional image filters such as blur, sharpen, etc.
- texture synthesis: the production of new textures from an image
- super-resolution: a higher-resolution image is obtained from a lower-resolution source
- texturization with an arbitrary image artistic filters and painting styles (e.g. impressionism, oil painting etc., see Fig. 17)
- texture-by-numbers: areas of a source image are annotated with colors. Through a simple painting interface, the artist now has the opportunity to create new images which resemble the old one (Figure 18).

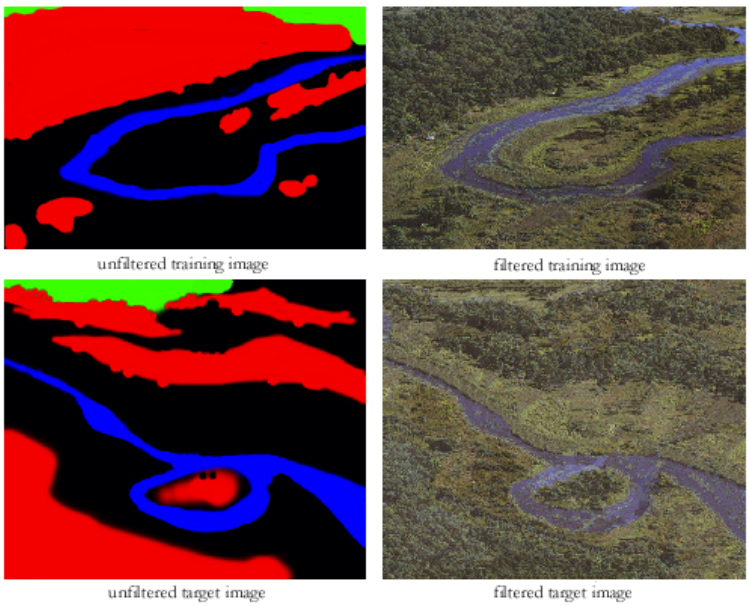


Figure 18: From a real landscape image, a new one is created by specifying the areas that should be copied into the target image

3.2 Painterly Rendering for Animation and Interaction

Figure 19: A graftal is an object which is used as a particle

Kaplan et al. [Matthew Kaplan '00] have proposed a **interactive artistic system** that is capable of producing virtually every drawing style *on 3D objects*. The approach physically simulates the distribution of particles on a surface. The particles themselves are **graftals** ([Smith '84, Reeves, Blau '85], see Figure 19), furry little triangle strips that align with the curvature of the surface. When used as particles, these graftals can produce typical stroke shapes formed by pencil, pen or brushes. They are aligned with the curvature of the object in order to gain the painted appearance. Frame-to-frame coherence is maintained by drawing all graftals for each frame. Possible uses of graftals for the simulation of fur and vegetation was discussed in [Kowalski et al. '99] (also see Figure 20).

Another approach which also uses particles to paint 3D objects, but is more artistically inspired than the former one is "**Painterly Rendering for Animation**" (Barbara J. Meier, [Meier '96]). Instead of rendering the painting in one step, the image is slowly made up of layers. The main idea of the approach is to describe surfaces by particles, depth-sort them with respect to the camera and render them as 2D brush strokes in screen space. The artist can influence color, size and orientation of the brush strokes.

Figure 20: A furry object using graftals

Figure 21 gives an overview of the *rendering pipeline* used: First, the particles need to be distributed on the surface. For this purpose, the surface is tessellated, and on each of the created triangles a random number of particles is put. These particles are then transformed to view space, and are rendered as brush strokes in screen space. The user can specify the appearance of each brush stroke using so-called reference images. The first one is for the *color* of the brush. The second one gives the *orientation*, and the third determines the *density* of the strokes that need to be painted on the object. A painterly renderer takes this information and computes an output image. For a hand-crafted look, the parameters of each brush stroke need to be randomized to a certain extent (for example, the orientation of the brush-strokes could vary slightly over the surface, see Figure 22).

As most valuable extension to the approach, the author has introduced a technique that first creates a rough *underpainting* and then refines this image with fine brush strokes. Furthermore, each 3D object is rendered as a separate layer, making it possible to adjust the painting parameters individually per object. The layers are then composed into the final image (**progressive painting**). In Figure 23, the haystack and the ground are rendered with different brush strokes. Furthermore, the painting is slowly produced from a dark and rough underpainting by consecutively adding more detail.

4 Cartoon-Style renderings

Cartoons simplify shapes and give a wide audience the ability to immerse into a simple, yet powerful form of art. Various papers exist on the topic, mostly dealing with the application of 2D computer graphics to cel animation [Durand '91, Litwinowicz '91, Fekete et al. '95]. This includes work on the so-called "**Multiperspective Panoramas**" [Wood et al. '97], which are flat images that give the illusion of a camera traveling through 3D scene (see Figure 24).

Other approaches deal with the use of computer graphics for the *improvement* of cartoon drawings. Examples include a semi-automatic generation of **shadows for**

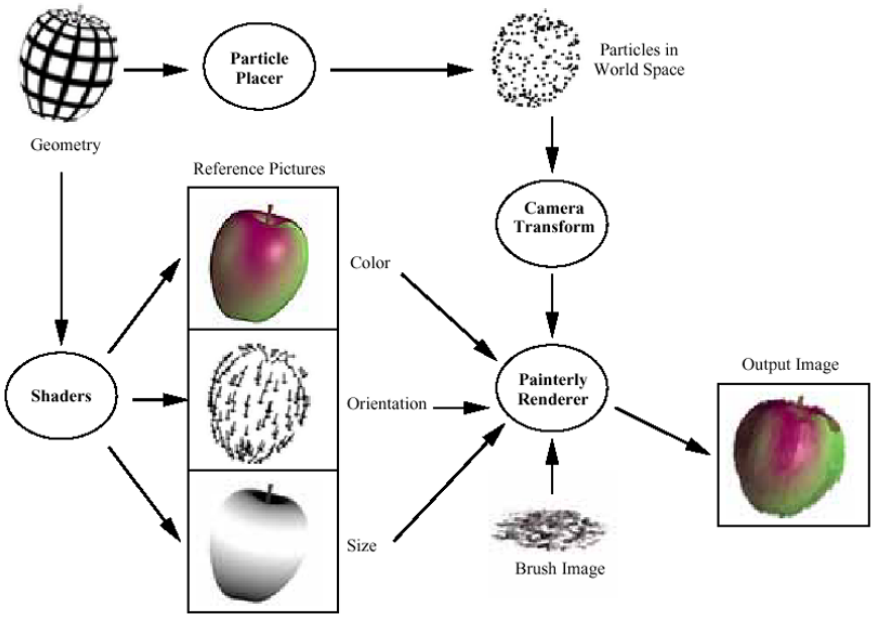


Figure 21: The pipeline for a painterly renderer

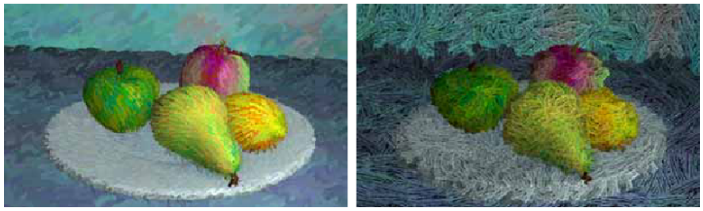


Figure 22: Hand-crafted look after applying randomness to the stroke orientation

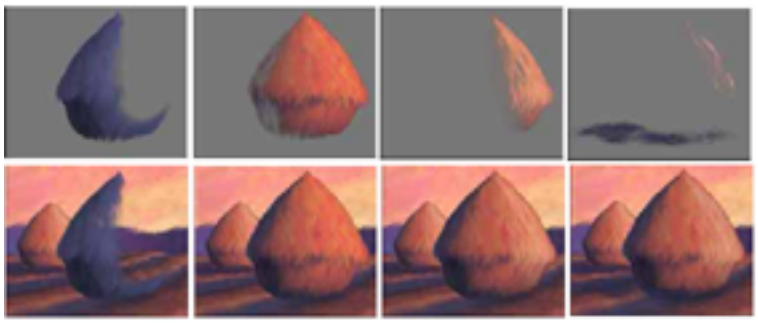


Figure 23: Progressive painting

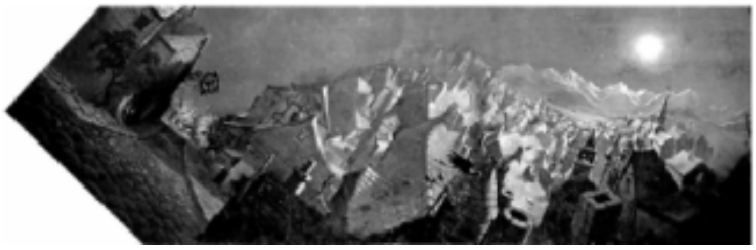


Figure 24: A multiperspective panorama from the Film *Pinochio*

cartoon scenes [Petrovic et al. '00] and the **application of complex textures to hand-drawn objects** [Corrêa et al. '98]. One of the first approaches with which cartoon rendering could be brought into the third dimension was the **shading model by Gooch et al.** [Gooch et al. '98]. Although not solely meant for use as cartoon shader, most approaches that try to render colored objects in a more non-photorealistic style use this color model. The key to this approach is the introduction of gradients into the shading model. Furthermore, edges are outlined so as to increase the illustrative look (see Figure 25).

A complete framework for generating **cartoon-style renderings in real-time** was presented by Lake et al. [Lake et al. '00]. The rendering process is divided into two parts: The *painter* determines the shading information to fill the polygons, and the *inker* highlights visible silhouette edges. Heart of the implementation is a new cartoon shading model that uses solid colors which do not vary over the materials they represent (see Figure 26). Shading is done with a color that is a darker version of the main material color. Technically, the approach does not smoothly interpolate shading across a model as in Gouraud shading, but finds a transition boundary and shades each side of the boundary with a solid color.

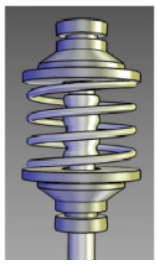


Figure 25: The Gooch Lighting Model gives a more descriptive look to objects by using gradients

The cartoon framework also embodies a pencil-style renderer, as well as the priorly mentioned inker, which detects and paints the silhouette edges of the mesh. Furthermore, **motion-lines** (lines giving hint of motion) are used to further emphasize animation. Another aspect of cartoon animation are the **view-specific distortions** of the models [Rademacher '99] (see Figure 27). There is a deformation on the 3d mesh which cannot be captured with conventional 3d models. Hence, the model has to change its shape autonomously with respect to the eye point.

Figure 26: A mesh rendered in cartoon style

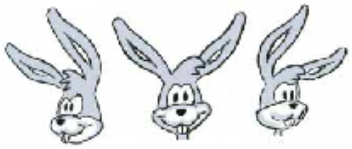
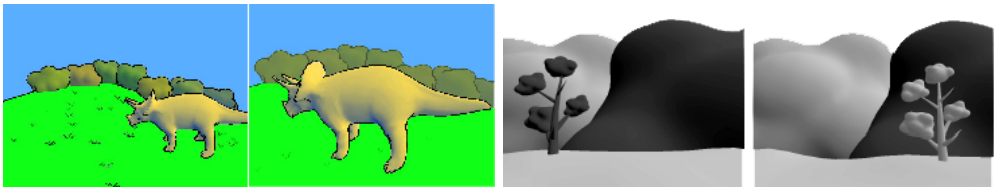


Figure 27: The ears of the mesh deform instead of just rotating with the head

Another similar idea presented by Rubin et al. [Kowalski et al. '01] is to let objects be aware of their *importance*: Less important items in the scene fade into the background, and are visually grouped together with other objects in order to reduce their prominence (see Figure 28). On the other hand, important objects stand out of the background and are thus emphasized (Figure 28). This process of selecting areas of high interest in a piece of art is widely known as **composition**.



(a) Distant objects are abstracted to form a single shape. The color is averaged over all objects.

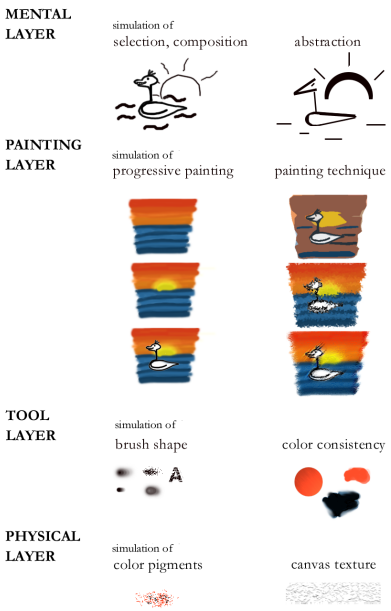
(b) An example of counterchange - the tree adjusts its color to contrast with the background

Figure 28: Composition

5 Conclusion

The intuitive categorization of computer-generated artwork used in this paper was based on the type of artwork (**pen-and-ink illustration**, **painterly rendering** or **cartoon-style**). A more general categorization would be dependent on the way in which the paintings are produced (refer to Figure **??**): At low level, there are approaches that try to physically simulate the distribution of pigments over a canvas (**physical layer**). The next layer, which I have called the **tool layer**, deals with the properties of drawing tools and colors. However, it does not care about the process of drawing itself. This is the subject of the next layer, called the **painting layer**. The **mental layer**, which forms the highest level in this model, looks at the way in which artists *think* when painting.

At the highest level (**mental layer**), we have seen an approach that deals with the problem of *composition* (Section 4, Cartoon-Style Rendering). In the **painting layer**, traditional painting style approaches for pen-and-ink were presented (see Section 2), starting with the *Comprehensible rendering of 3D-Shapes* and *Computer-Generated Pen-and-Ink Illustrations*. With the use of fast *silhouette determination* and *Tonal Art Maps,* it is now possible to render pen-and-ink-style drawings in *real-time,* for example in *games*. Other types of painting styles mentioned in this report include the rendering of *Chinese Portrait and Landscape Paintings*, *Sumi-e* (Section 2.2) and *Cartoon-Style Images* (*Section 4*). Additionally, an approach that simulates the artistic learning of painting style through repetition (*Image Analogies*, Section 3.1) was presented. For the **tool layer**, we have seen a simulation of *painting with various brush strokes and sizes* (Section 3.1) as well as a discussion on the influence of *line direction* (Section 2.1.2) for hatched illustrations. Colors are often simulated in the **physical layer**, with *watercolor* (Section *3.1*) and *graphite pencils* (Section 2.1.2) being examples. Furthermore, the mentioned *three-dimensional painting techniques* (Section 3.2) can also be seen as physical approaches, because the distribution of particles on a surface is computed.



References

Michael Gleiche Alex Mohr (2001). Non-invasive, interactive, stylized rendering. In *Proceedings on 2001 Symposium on Interactive 3D graphics March 2001 Pages: 175 - 178 Series-Proceeding-Article Year of Publication: 2001 ISBN:1-58113-292-1*, 2001.

Arthur Appel (1967). The notion of quantitative invisibility and the machine rendering of solids. In *Proceedings of the twenty second ACM national conference 1967, pages 387-393*, S. 387–393. ACM, 1967.

Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, Adam Finkelstein (1998). Texture Mapping for Cel Animation. In **Michael Cohen** (Hrsg.) (1998), *Proceedings of SIGGRAPH 98*, Annual Conference Series, Addison Wesley, S. 435–446. Addison Wesley, 1998.

Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, David H. Salesin (1997). Computer-generated watercolor. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques August 1997 Pages 421 - 430 Series-Proceeding-Article Year of Publication 1997 ISBN:0-89791-896-7*, 1997.

Debra Dooley, Michael F. Cohen (1990). Automatic illustration of 3D geometric models: lines. In *Proceedings of the 1990 symposium on Interactive 3D graphics, pages 77-82*, S. 77 – 82. ACM, 1990.

Charles Durand (1991). The Toon Project: Requirements for a Computerized 2D Animation System. S. 285–293. ACM, 1991.

Jean-Daniel Fekete, Erick Bizouarn, Eric Cournarie, Thierry Galas, Frederic Taillefer (July 1995). TicTacToon: A Paperless System for Professional 2D Animation. S. 79–90. ACM, July 1995.

David H. Salesin Georges Winkenbach (1994). Computer-generated pen-and-ink illustration. In *Proceedings of the 21st annual conference on Computer graphics July 1994 Pages: 91 - 100 Series-Proceeding-Article Year of Publication: 1994 ISBN:0-89791-667-0*, 1994.

Ahna Girshick, Victoria Interrante, Steven Haker, Todd Lemoin (2000). Line direction matters. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering June 2000 Pages: 43 - 52 Series-Proceeding-Article Year of Publication: 2000 ISBN:1-58113-277-8*, 2000.

Amy Gooch, Bruce Gooch, Peter Shirley, Elaine Cohen (July 1998). A Non-Photorealistic Lighting Model for Automatic Technical Illustration. In **Michael Cohen** (Hrsg.) (July 1998), *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 1998), pages 447-452*, Computer Graphics Proceedings, Annual Conference Series, S. 447–452. ACM SIGGRAPH, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

Bruce Gooch, Peter-Pike J. Sloan, Amy Gooch, Peter Shirley, Richard Riesenfeld (1999). Interactive Technical Illustration. In *Proceedings of the 1999 symposium on Interactive 3D graphics, pages 31-38*, S. 31 – 38, 1999.

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, David H. Salesin (2001). Image Analogies. In **Eugene Fiume** (Hrsg.) (2001), *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 327–340. ACM Press / ACM SIGGRAPH, 2001.

Aaron Hertzmann, Ken Perlin (2000). Painterly rendering for video and interaction. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering June 2000 Pages: 7 - 12 Series-Proceeding-Article Year of Publication: 2000 ISBN:1-58113-277-8*, 2000.

Aaron Hertzmann. (1998). Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In **Michael Cohen** (Hrsg.) (1998), *Proceedings of SIGGRAPH 98*, Annual Conference Series, Addison Wesley, S. 453–460. Addison Wesley, 1998.

Aaron Hertzmann (2001). Painting By Relaxation. In *CGI 2001*, 2001.

Walter Koschatzky (1990). *Die Kunst der Zeichnung.* Edition Atlantis, Grafische Sammlung Albertina. ISBN: 3-88199-725-3.

Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden, John Hughes (1999). Art-Based Rendering of Fur, Grass, and Trees. In **Alyn Rockwood** (Hrsg.) (1999), *Siggraph 1999, Computer Graphics Proceedings*, Annual Conference Series, S. 433–438, Los Angeles, 1999. ACM Siggraph, Addison Wesley Longman.

Michael A. Kowalski, John F. Hughes, Cynthia Beth Rubin, Jun Ohy (2001). User-guided composition effects for art-based rendering. In *Proceedings on 2001 Symposium on Interactive 3D graphics March 2001 Pages: 99 - 102 Series-Proceeding-Article Year of Publication: 2001 ISBN:1-58113-292-1*, 2001.

Adam Lake, Carl Marshall, Mark Harris, Marc Blackstein (2000). Stylized Rendering Techniques For Scalable Real-Time 3D Animation. ACM, 2000.

Peter Litwinowicz (July 1991). Inkwell: A 2.5-D Animation System. S. 113–122. ACM, July 1991.

Peter Litwinowicz (1997). Processing images and video for an impressionistic effect. ACM, 1997.

Aditi Majumder, M. Gopi (2001). Hardware Accelerated Real Time Charcoal Rendering. ACM, 2001.

Lee Markosian, Michael A. Kowalski, Daniel Goldstein, Samuel J. Trychin, John F. Hughes, Lubomir D. Bourde (1997). Real-time nonphotorealistic rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques August 1997 Pages: 415 - 420 Series-Proceeding-Article Year of Publication: 1997 ISBN:0-89791-896-7*, 1997.

Elaine Cohe Matthew Kaplan, Bruce Gooch (2000). Interactive artistic rendering. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering June 2000 Pages: 67 - 74 Series-Proceeding-Article Year of Publication: 2000 ISBN:1-58113-277-8*, 2000.

Barbara J. Meier (1996). Painterly rendering for animation. In *Proceedings of the 23rd annual conference on Computer graphics August 1996 Pages: 477 - 484 Series-Proceeding-Article Year of Publication: 1996 ISBN:0-89791-746-4*, 1996.

Ramesh Raskar and Michael Cohen (1999). Image precision silhouette edges. In *Proceedings of the 1999 symposium on Interactive 3D graphics, pages 135-140*, S. 135 – 140. ACM, 1999.

J. D. Northrup, Lee Markosian (2000). Artistic silhouettes. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering June 2000 Pages: 31 - 37 Series-Proceeding-Article Year of Publication: 2000 ISBN:1-58113-277-8*, 2000.

Lena Petrovic, Brian Fujito, Lance Williams, Adam Finkelstein (2000). Shadows for Cel Animation. ACM, 2000.

Emil Praun, Hugues Hoppe, Matthew Webb, Adam Finkelstein (2001). Real-Time Hatching. In **Eugene Fiume** (Hrsg.) (2001), *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 579–584, 2001.

Paul Rademacher (1999). View-Dependent Geometry. In **Alyn Rockwood** (Hrsg.) (1999), *Siggraph 1999, Computer Graphics Proceedings*, Annual Conference Series, S. 439–446, Los Angeles, 1999. ACM Siggraph, Addison Wesley Longman.

William T. Reeves, Ricki Blau (1985). Approximate and probabilistic algorithms for shading and rendering structured particle systems. ACM, 1985.

I. H. H. Lee S. C. Hsu (1994). Drawing and Animation Using Skeletal Strokes. ACM, 1994.

Takafumi Saito, Tokiichiro Takahashi (1990). Comprehensible Rendering of 3-D Shapes. In **Forest Baskett** (Hrsg.) (1990), *Computer Graphics (SIGGRAPH '90 Proceedings) vol. 24, no. 4, pages 197 - 206*, Volume 24, S. 197–206, 1990.

Michael P. Salisbury, Michael T. Wong, John F. Hughes, David H. Salesin (1997). Orientable textures for image-based pen-and-ink illustration. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques August 1997 Pages: 401 - 406 Series-Proceeding-Article Year of Publication: 1997 ISBN:0-89791-896-7*, 1997.

Jutta Schumann, Thomas Strothotte, Andreas Raab, Stefan Laser (1996). Assessing the Effect of Non-Photorealistic Rendered Images in CAD. ACM, 1996.

Alvy Ray Smith (1984). Plants, Fractals, and Formal Languages. ACM, 1984.

Mario C. Sousa, John W. Buchanan (1999). Computer-Generated Graphite Pencil Rendering of 3D Polygonal Models. In **P. Brunet, R. Scopigno** (Hrsg.) (1999), *Computer Graphics Forum (Eurographics '99)*, Volume 18(3), S. 195–208. The Eurographics Association and Blackwell Publishers, 1999.

S. Strassmann (1986). Hairy Brushes. ACM, 1986.

L. Streit, J. Buchanan (1998). Importance Driven Halftoning. In **David Duke, Sabine Coquillart, Toby Howard** (Hrsg.) (1998), *Computer Graphics Forum*, Volume 17(3), S. 207–217. Eurographics Association, 1998.

Thomas Strothotte, Bernhard Preimand Andreas Raab, Jutta Schumann, David R. Forsey (1994). How to Render Frames and Influence People. ACM, 1994.

Der-Lor Way, Chih-Wei Hsu, Hsin-Yi Chiu, Zen-Chung Shih (2001). Computer-Generated Chinese Painting for Landscapes and Portraits. In **V. Skala** (Hrsg.) (2001), *WSCG 2001 Conference Proceedings*, 2001.

Georges Winkenbach, David H. Salesin (1996). Rendering Parametric Surfaces in Pen and Ink. In *Proceedings of the 23rd annual conference on Computer graphics 1996, pages 469-476*, S. 469 – 476. ACM, 1996.

Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, David H. Salesin (1997). Multiperspective Panoramas for Cel Animation. In *SIGGRAPH 97*, 1997.