Non-Photorealistic Animation and Rendering
Pierre Bénard and Holger Winnemoeller (Editors)

Art-directed Watercolor Rendered Animation

S. E. Montesdeoca[†1], H. S. Seah[1], H.-M. Rall[1],

[1]Nanyang Technological University, Singapore

Figure 1: Close-up of an art-directed watercolor simulated render. Henry © Oculus Story Studio

Abstract

This paper presents a system to render 3D animated geometry as watercolor painted animation with art-directed control. Our approach focuses on letting the end user paint the influence of the modeled watercolor effects in the 3D scene to simulate the characteristic appearance of traditional watercolor. For this purpose, it performs an object-space simulation and makes use of the user-painted influences to control and enhance image-space watercolor effects. In contrast to previous approaches, we introduce specialized watercolor shaders that are adjusted and deformed according to the desired painted effects. We further present novel algorithms that simulate hand tremors, pigment turbulence, color bleeding, edge darkening, paper distortion and granulation. All of these represent essential characteristic effects of traditional watercolor. The system performs in real-time, scales well with scene complexity and is fully implemented in Autodesk Maya.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations I.3.4 [Computer Graphics]: Graphics Utilities—Paint systems I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.m [Computer Graphics]: Miscellaneous—Non-photorealistic Rendering I.4.3 [Computer Graphics]: Enhancement—Filtering

1. Introduction

The characteristic appearance and effects found on watercolor paintings make them a unique traditional medium which is extensively used for multiple coloring purposes. From early topological maps to fashion design illustrations and fine art. As Leslie Dutcher describes it, "watercolor paint has proven to be as elegant and academic as its counterpart [oil] while retaining its own ebullience and congeniality" [RD13].

However, while regarded as the medium by itself is, it is rarely seen in animated form. The cause for this probably has to do with the accompanying difficulty to control the medium. Even acclaimed watercolor artists admit the capricious nature of the medium, which makes it difficult to master [Ong03, Sch14, Pol13]. This peculiarity, together with the need for at least 10-12 frames per second to entail a sense of motion [RM00], makes traditional watercolor animation a complex and laborious endeavor to pursue.

Various systems to digitally recreate watercolor have been proposed, which help alleviate the production of traditional watercolor

† santiago002@ntu.edu.sg

paintings and animations. These systems can be organized into three main categories according to their respective input data:

- Stroke-space systems
- Image-space systems
- Object-space systems

Each category has its own specialized application, however, they often tend to interrelate in order to enhance the final watercolor simulation.

The system proposed in this paper is modeled to render 3D animated geometry as watercolor simulated animations, which fall into the third category, an object-space system. However, it differentiates itself from other object-space systems by two key elements.

First, the proposed system is art-directed by the end user who can paint the modeled effects as desired. This feature enables immediate localized control over the watercolor look, with real-time performance. And second, the proposed system extends the palette of simulated watercolor effects and does not require previously painted watercolor images. Additionally, the proposed system introduces the following contributions:

- Specialized shaders with a watercolor reflectance model, object-space hand tremors and custom object-space features, which enhance and control further image-space filtering.
- Pigment turbulence in object-space (art-directed).
- Color bleeding through a hybrid solution involving object- and image-space processing (art-directed).
- Edge darkening with gradual pigment accumulation through difference of Gaussians feature enhancement (art-directed).
- Paper distortion through normal gradients (art-directed).
- Split paper granulation filter enabling dedicated control over paper roughness and pigment accumulation at the valleys of the paper (art-directed).

Figure 1 shows a close-up obtained from a 3D model, which shows most of the watercolor effects simulated by our system. Additional rendered images can be found throughout the paper. For animated outcomes, please see the accompanying audiovisual material. An abstracted pipeline schematic can be found in Figure 2.

In the following, related work will be introduced in Section 2. The proposed system with its individual contributions is presented in Section 3 and the results are presented and discussed from Section 4 onward.

2. Related Work

Recreating traditional painting media has been one of the main challenges of non-photorealistic rendering since its inception. Remarkable examples came early on through physical approximation of the medium which analyzed and calculated the spread of pigment in simulated paper [Sma91, CAS*97].

Stroke-space simulation systems especially benefited from accurately modeled physical approximation approaches to watercolor. Each brush-stroke can be intricately controlled to achieve most desired effects and a characteristic watercolor look [YJC*13, VLVR05, CT05]. Nonetheless, it can be difficult for an artist to understand and interpret the physical attributes which these simulations offer. Alternatives can be found through procedural and example based approaches [DKMI13, LBDF13]. Stroke-space watercolor simulations provide extensive artistic control over digital paintings, however, in animation, it still presents itself as a complex and laborious alternative where each frame requires to be hand painted.

For the proposed system, stroke-space simulations can be used in order to paint watercolor-looking textures which are then mapped into 3D objects. However, these textures would remain static and even distort without further processing. Additionally, the aim of the proposed system is to not rely on previously painted textures for a watercolor-look. Related work in image-space and object-space simulations are directly relevant to our system and will be presented next.

2.1. Image-space simulation

A traditional watercolor painting is a composition of a variety of complex interactions of pigment and water, layered together to create a body of work which interacts and speaks for itself with unique attributes, all chosen by the artist. Image-space simulation attempts to recreate this interaction and considers the image as a whole - finding and simulating characteristic effects and features which the artist might have chosen with the given subject.

Early attempts use image processing algorithms to convert raster images to watercolor simulated paintings. Johan et al. [JHN04] focused on recreating watercolor brush strokes by following a vector field created along the boundaries of the image elements. Bousseau et al. [BKTS06] focused instead on recreating the wet-on-dry (lavis) technique using a series of filters. Characteristic effects such as turbulent pigment flow, pigment dispersion, edge darkening, paper structure and its distortion on the painted subject were considered. However, both of these methods process the image as a whole, limiting control of the outcome to variables and predefined rules.

The system proposed by Wang et al. [WWF*14] considers this limitation and involves computer vision algorithms to automatically locate the area - which artists might emphasize - through saliency detection. Additionally, the input image is colorized according to colors found on a training set of real watercolor paintings. The palette of simulated watercolor effects was also extended to include color bleeding found on wet-in-wet techniques and hand tremors which are common in most watercolor paintings.

However, the previously introduced systems rely heavily on image segmentation to abstract detail from the input data. Segmentation over time introduces significant flickering issues, which are not a desirable effect in watercolor animations. Image-time-space simulation of watercolor has been proposed to address this issue by Bousseau et al. [BNTS07] but it still produces unnatural artifacts and is limited in the amount of effects it can simulate.

Finally, example based approaches have also been proposed in order to simulate watercolor animation using image analogies. Fišer et al. [FLJ*14] proposed a system to simulate noise from static watercolor textures. Bénard et al. [BCK*13] also implements

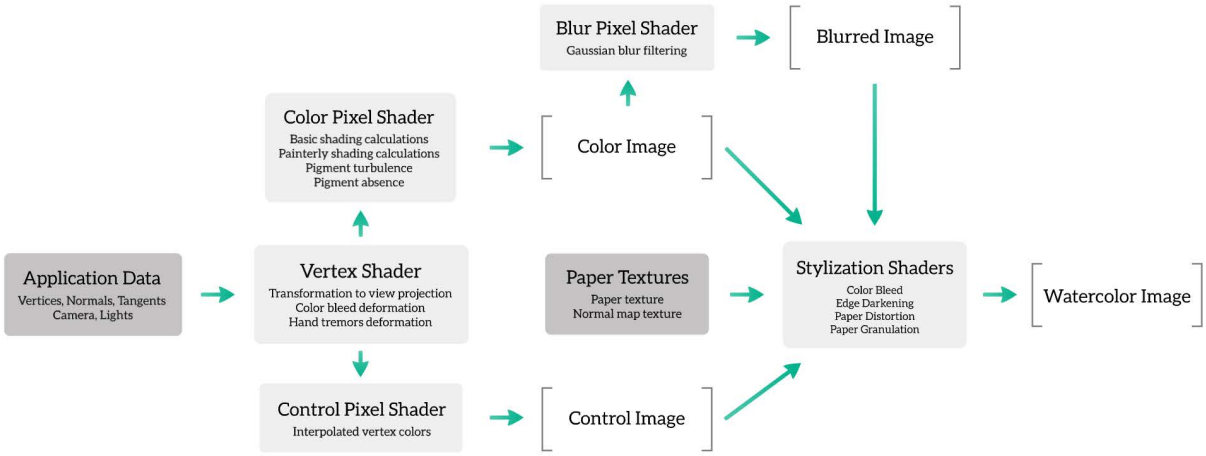


Figure 2: Current system schematic portraying the rendering pipeline. Dark gray elements represent the input data, gray elements represent the different processing stages and the bracketed elements represent the computed images in the frame buffer.

image analogies to interpolate painted keyframes done by artists. This is the only system that offers an art-directed watercolor stylization in which the user has localized control over the final look. However, as textures are required to be painted every few frames, it cannot be used in interactive applications where the point of view is not previously defined, such as games and virtual reality.

2.2. Object-space simulation

Object-space simulation approaches the characteristic effects of watercolor in a more localized manner, as each 3D object will need to perform the stylization by itself. However, most object-space simulations are conjugated with image-space simulations to enhance the outcome.

The system proposed by Lum and Ma [LM01] was one of the first to simulate watercolor effects in object-space by incorporating pigment turbulence through the use of filtered noise along the curvature of the geometry. Burgess et al. [BWK05] extends this work by simplifying the noise filter used for pigment turbulence and using object identifiers and depth to find darkened edges and incorporate edge distortion through additional noise. However, the results of both watercolor simulated systems offer limited effects, control and resemblance to the traditional counterpart.

Lei and Chang [LC05] proposed a programmable vertex and fragment shader system which included a custom reflectance model based on a simulated color ramp. Image-space simulations, such as edge darkening through edge detection and paper effects to recreate granulation and distortion, were performed afterwards. Finally, Luft and Deussen [LD06] proposed a complex layered system where each object with a common identifier is rendered in multiple passes and each pass is processed by image-space simulations to later be composed into a final image - together with all other objects. Further particles are also attached to the geometry in the case of trees to simulate leaves. The system improves on art-directed control as each object group gets simulated independently. It also features paper effects such as distortion and granulation, improves on edge darkening and adds abstraction levels. However, the system does not scale well with scene complexity and still misses fundamental characteristics such as pigment turbulence.

All modeled object-space systems to simulate watercolor offer a limited reproduction of characteristic effects and basic art-direction. We propose a system, which contributes to the field, by improving on these two key elements and scales well with scene complexity.

3. Art-directed Watercolor System

The aspiration of our system is to create a powerful, user-friendly object-space watercolor simulation tool that not only focuses on modeling the characteristic effects of watercolor, but also considers real-time performance and art-direction as a necessary requirement. In the following, the art-directed object-space control over the watercolor outcome is explained (Section 3.1). Then, the system portrayed in Figure 2 is deconstructed and the modeled effects presented. The currently supported effects include hand tremors (Section 3.2), color bleeding (Wet-in-Wet, Sections 3.2 and 3.3), pigment turbulence, edge darkening and paper effects such as distortion and granulation (Section 3.3).

3.1. Art-direction

Artists will usually not paint an entire painting with an even treatment. They will emphasize, abstract, and add custom effects according to their own style and expression. This is especially true with the aesthetic quality of watercolors, which permits a wide variety of expressions through their volatile effects.

For this purpose, we adopted the masking technique commonly used for image-space manipulation. However, our mask is not created from painted pixels, but by vertices (points) in 3D space. Alex Harvill [Har07] has also implemented this idea by rendering a map

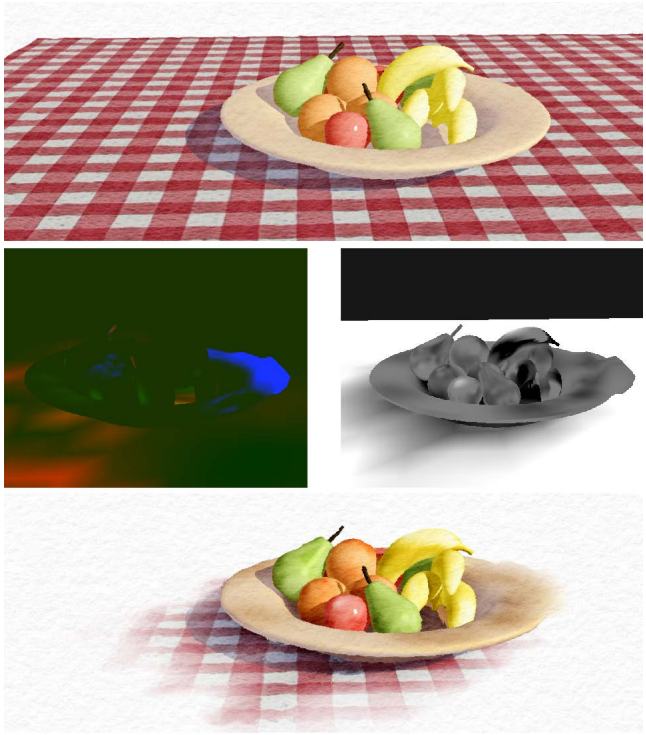


Figure 3: Top to bottom: color image rendered without art-direction; control image showing vertex colors (Left: RGB channels | Right: Alpha channel); art-directed watercolor simulation.

of weighted points to control a moving 2D illustration. From custom texture maps to point weights and vertex colors, these scalar field data can be used in additional render passes as masking devices for later image-space simulations.

In our system, we have chosen vertex colors for multiple reasons. They provide four control channels (RGBA), are natively supported by vertex shaders (compute fast), can be animated for further temporal control and are supported by most 3D software vendors.

For the implementation presented in this paper, we have assigned the control of effects to the following channels:

- Red -> Controls paper distortion
- Green -> Controls paper granulation
- Blue -> Controls edge darkening and color bleeding
- Alpha -> Controls pigment turbulence and pigment fading

A deconstructed example of the art-directed influences and the dramatic change it has over the 3D rendered outcome can be seen in Figure 3.

3.2. Controlled Object-space Effects

The first step in the rendering pipeline consists of processing the application data at the vertex shader stage. At this stage, normally the vertices are transformed from object space to the respective projection space. However, our simulation involves two additional distinct vertex deformations - for simulation purposes - which are explained next.

Wet-in-Wet and Hand Tremor Deformation

The Wet-in-Wet technique, also referred to as color bleeding, is a technique where pigment is applied to an already wet surface to allow the color to spread (bleed) outside of the placement area.

In order to simulate the color bleeding effect, the vertices, which have been assigned the respective vertex color, are translated along their normal vector. This is done in order to pull the vertices outside of its original geometry, simulating the pigment bleeding outside its contained area. However, the vertices are translated in different proportions for the control image and the color image ($control = 2.5 \times color$). The chosen vertex offsets enable a better result later on down the pipeline in Section 3.3.

Hand tremors are an essential characteristic of hand painted media as acknowledged by Wang et al. [WWF*14]. They are observable in the small irregularities found mostly on the edges of a painting and are caused by involuntary fluctuations in the human nervous system.

These fluctuations can be simulated by minimally offsetting the vertices from the geometric objects, once they are transformed to the projection space. We chose a sinusoidal noise function as it can additionally simulate watery deformation, if desired.

$$V_o = sin(T \times s + V \times f) \times t \times P_p \tag{1}$$

The vertex offset V_o is modulated by the original vertex position V, time T and relative pixel size of the projection space P_p. Adjustment variables for speed s, frequency f and tremor amount t can be chosen for each shaded object - for any desired tremor effect. However, hand tremors occur mostly in edges where precise control is required by the artist. Therefore, the vertex tremor V_t is modulated to do so with the dot product of the view direction $\vec{V}$ and vertex normal $\vec{N}$.

$$V_t = V + V_o(1 - a(\vec{V} \cdot \vec{N})) \tag{2}$$

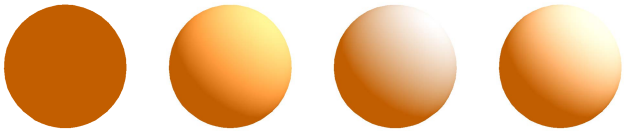
Once the geometry has been deformed and the additional necessary data calculated, the pixel shader stage calculates the color and control image.

Watercolor Reflectance Model and Pigment Turbulence

The characteristic translucency found in watercolors plays a pivotal role in the way motifs are colored, but is not found in common shading primitives. Lei and Chang [LC05] proposed the use of a one dimensional color ramp to define which colors are reflected throughout the surface. However, this presents problems if the user intends to use texture maps. Therefore, we propose a custom shading model based on common shading primitives with additional watercolor characteristics such as pigment dilution, cangiante illumination (change in the highlights to a brighter hue) and art-directed pigment turbulence, together with some minor additional painterly effects.

To simulate dilution, we calculate the area of effect D_A by modulating the dot product of the fragment normal $\vec{N}$ and the light direction $\vec{L}$ with a dilute area variable $d_A \in (0,1]$.

$$D_A = \frac{\vec{L} \cdot \vec{N} + (d_A - 1)}{d_A} \tag{3}$$



$c = 0, d = 0$ $c = 0.6, d = 0$ $c = 0, d = 1$ $c = 0.7; d = 0.8$

Figure 4: Watercolor shader with different cangiante and dilution attributes ($d_A = 1; C = (194, 94, 0)$)

Once the area is computed, the surface color may undergo the changes of a cangiante illumination which shifts the hue of the surface away from a primary color. Through color theory in art and human perception, it is known that highlights do not necessarily shift perceptively towards white. The same rules apply to shadows which also do not necessarily shift perceptively towards black [Bir87]. To achieve this, the cangiante color C_c is calculated by adding the dilute area intensity multiplied by the cangiante variable c.

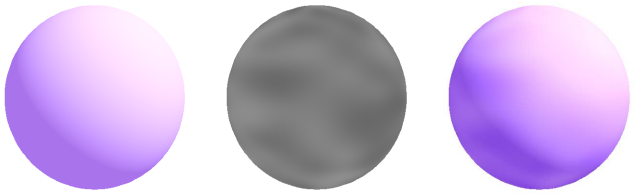
$$C_c = C + (D_A \times c) \tag{4}$$

Once the hue at the highlight has shifted, the dilution is performed by linearly interpolating the cangiante color towards the paper color C_p through the dilution variable d.

$$C_d = d \times D_A(C_p - C_c) + C_c \tag{5}$$

Results of the shader with different attributes is shown in Figure 4.

Pigment turbulence is a low-frequency noise which happens when the pigment density is placed unevenly over the flat surface of the paper. This effect is characteristic for watercolors and is preferably simulated in object-space. Image-space simulations of pigment turbulence have been implemented before, however they are overlaid across the entire image. Therefore, it does not distinguish between different painted areas and present "shower-door" artifacts.



Without turbulence Painted turbulence With turbulence

Figure 5: Shader turbulence breakdown ($C = (169, 115, 235)$).

In our approach, we let the user art-direct the density variations to suit the desired amount and pattern by painting the vertex control channel $Ctrl \in [0, 1]$ directly on the object within the 3D application. An example of this and the influence it has over the rendered geometry can be seen in Figure 5. In order to simulate the accumulation of pigments, the color transmittance modification model, proposed by Bousseau et al. [BKTS06], is used. In this model, the observed color is the result of the exponential attenuation of reflected light which has been transmitted through the density of the pigment. We additionally use the same control channel to fade the rendered color C towards the paper color C_p. This is a common stylistic choice found in watercolor illustrations where the painter avoids unnecessary scene elements by simply fading the color and leaving the paper, as is. An example of this effect can be seen in Figure 3 at the tablecloth. Both algorithms can be found in the equation below.

$$C_t = \begin{cases} C^{3-(Ctrl \times 4)} & \text{if } Ctrl < 0.5 \\ (Ctrl - 0.5) \times 2(C_p - C) + C & \text{if } Ctrl \geq 0.5 \end{cases} \tag{6}$$

To enhance the look of the shader, we support basic shading requirements like the ability to use texture, normal and specular maps. Additionally, painterly reflectance attributes were also implemented, such as a custom diffuse shading factor, custom shade color, shade wrap, specularity, hard highlights and atmospheric fade. However, the deconstruction of these features have been omitted as they are not necessary characteristics of watercolors.

After the reflectance is calculated, the resulting color and control image are stored in the frame buffer and are processed further by image-space simulations to obtain the final watercolor image.

3.3. Controlled Image-space Effects

Before entering the image-space stylization stage, additional paper textures, together with a low-pass filtered color image, are required. The paper textures involve a grayscale intensity map with the structure of the paper and its respective normal map, containing the surface inclinations in the red and green channels. The low-pass filtered color image is calculated by convolving a 21x21 gaussian kernel ($\sigma = 20$) over the color image found at the frame buffer. To amplify the filtering on minimal computational cost, the blurred image is calculated at a render target, half the size as the other images, and linearly interpolated through a texture sampler at later stages. Once these images have been loaded and calculated, the image-space stylization stage begins.

Color Bleeding

Previously in Section 3.2, the assigned vertices were pulled outside their geometry, according to the user painted influence. To finalize the color bleeding simulation, the blurred image I_b is masked to the color image I according to the control channel $Ctrl$ - which previously pulled out the geometry.

$$I_{cb} = Ctrl(I_b - I) + I \tag{7}$$

This is done in order to simulate the pigment diffusion which occurs when the colors bleed into each other. In our implementation, the blue vertex channel controls this effect and it can be seen in Figure 3.

Edge Darkening

Edge darkening is a characteristic effect which manifests itself when the pigment is carried out to the boundaries of a colored area due to surface tension. This causes a higher concentration of pigments at the edges of the image and, therefore, a darker appearance.

Edge darkening has been one of the most studied and implemented characteristic effects. Early solutions in object-space involved analyzing nearby pixels for a difference in depth and object

identifiers [BWK05] and subtracting a blurred intensity from the alpha values of object identifiers [LD06] - which in turn dilutes the main color. However, these models present a problem when texture information needs to be taken into account. Current solutions to simulate edge darkening in image-space involve edge detection algorithms such as the sobel filter [LC05, BKTS06]. However, edge darkening is not simply a darkened outline which would result from most edge detection algorithms. Edge darkening presents itself as a gradient darkening along the boundaries of the previously wet surface.

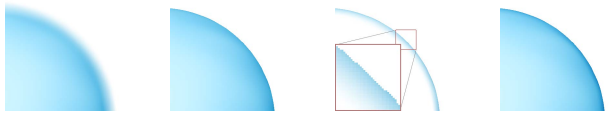


Figure 6: Edge darkening simulation process ($C = (31, 171, 224)$)

To model the gradient pigment accumulation found at the edges, we build upon the difference of Gaussians (DoG) feature enhancement algorithm. However, as the images do not contain any noise, only one low-pass filtered image is required. For this purpose, we take the previously calculated blurred image found in the frame buffer. Additionally, in our algorithm, the image I is subtracted from the blurred image I_b. This deviation from the standard difference of Gaussians algorithm is done in order to avoid edge darkening at brighter areas of the image such as the paper color. Once the edges have been calculated, the maximum intensity value from the RGB channels is taken and we use the color transmittance modification model to darken the color as seen in Equation 7. The results from our model can be seen in Figure 6.

$$I_{ed} = I_{cb}^{1+Ctrl \times max(I_b - I)} \tag{8}$$

Darkened edges do not occur evenly throughout the painting, therefore, the user is able to art-direct this effect by modifying the $Ctrl$ color value.

Paper Distortion and Granulation

Paper distortion occurs when the rough surface of the paper texture creates small distortions to the areas which are currently being painted. This happens through the interaction between the roughness of the surface and the hair of the brushes or through surface inclinations which move the pigments towards the valleys of the paper. The later cause is also directly correlated with the paper granulation effect. Paper granulation is the result of the higher concentration of pigments found at the valleys of the paper which generate a darkened appearance. In order to simulate both of these effects, actual paper textures were used, as implemented by previous researchers [BKTS06, WWF*14, LC05, LD06].

In our system, paper distortion is effectively computed by directly sampling the color values at UVs which have been shifted by the surface inclinations found at the normal map paper texture. The amount of paper distortion can also be art-directed towards any desired result.

The final step in our pipeline involves the paper granulation effect. Our model is done in a way that permits individual control over the roughness of the paper and the accumulation of the pigments at the valleys of the papers. One of the main drawbacks of the color transmittance modification model is that once a channel is fully saturated, the color will not be darkened as $1^n = 1$ for $n \in \mathbb{R}$. Therefore, a split model is proposed which linearly interpolates between the color transmittance modification model and color subtraction, depending on the image saturation I.

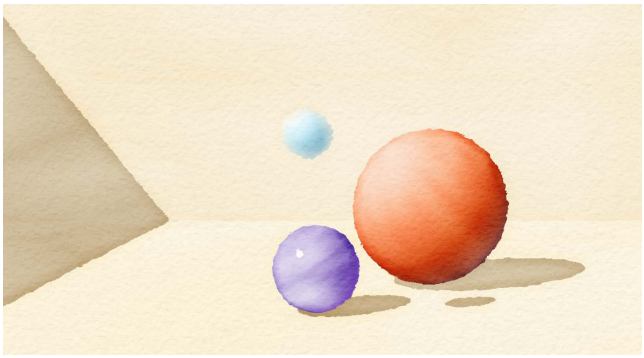
$$I_g = I(I - P_{iv}) + (1 - I)I^{1+(Ctrl \times dP_{iv})} \tag{9}$$

The density amount variable d modulates the intensity of the inverted paper texture P_{iv}. This in turn is controlled by a $Ctrl$ channel in order to art-direct the pigment density at the valleys of the paper.

4. Results and Discussion

Once the geometry has been through all the changes and calculations from Section 3 onward, the final watercolor simulated image is presented to the user.

Various rendered images can be seen throughout Figure 7. To observe a breakdown of the simulated effects, the system and the rendered simulation in animation, please refer to the accompanying video. An additional comparison to other object-space watercolor simulation systems can be observed in Figure 8.



(a) Spheres with different settings

(b) Gotita

(c) Henry

Figure 7: Rendered frames captured from the Maya viewport

Our system has been implemented in Autodesk Maya and uses Viewport 2.0 as its engine. The simulation scales well with scene complexity and performs in real-time with the following hardware configuration: Quadro k5000, Intel Xeon E5-2609 @2.40 and 16gb RAM. Technical details and performance statistics for each scene can be seen in Table 1. Real-time performance is crucial for art-directed systems as the user receives immediate feedback which dramatically enhances any iterative work-flows to achieve the desired result. Additionally, With the ability to model, animate, shade, light and render within the same software, pipeline overheads can be avoided.

Table 1: Performance on different scenes (resolution: 1080p)

Figs.	Triangles	FPS (avg)
3	83440	122
7a	26088	155
7b	57000	140
7c	106106	120

As it can be appreciated from the rendered images in Figure 7 and the comparisons shown in Figure 8, the results presented by our system offer a significant improvement in art-direction and, therefore, the overall characteristic look of watercolor. Some previous attempts have come close to recreate a specific style. However, watercolors can be used to create artwork in a variety of different ways. Therefore, we focused on enabling this control in order to reproduce this stylistic freedom. Thanks to this key feature, it could be possible for our system to recreate the results presented by previous researchers and to go beyond them to create unique graphics that come closer to any desired watercolor style.

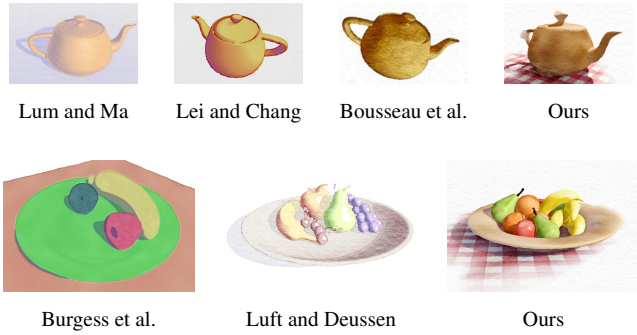


Figure 8: Comparison between watercolor simulation systems (magnify images to see details)

The watercolor characteristics we have modeled extend the palette of watercolor effects found in object-space systems to include hand tremors, color bleeding and a painterly watercolor reflectance model which takes advantage of the unique translucence and turbulence of the medium. We also enhanced existing effects by modeling edge darkening with gradual pigment accumulation, proposed an efficient way to calculate paper distortion and enabled split control over paper granulation. All these effects are complemented by art-directed localized control which completely transforms the rendered outcome - depending on the creative abilities of the user.

Limitations

An inherent limitation of our system lies on the vertex density of polygonal objects. As control is assigned per-vertex, a certain degree of polygonal resolution is required to guarantee proper control. Mindful modeling and different levels of detail are suggested to overcome this limitation.

Art-directed systems are also propense to overwhelm with control which might not necessarily be required. This in turn affects the performance and adds clutter to the control application. In our tests, users rarely used the localized control for granulation and distortion. For these effects, they were satisfied with global controls for the entire scene. Further study of user behavior is required to optimize the system and address this limitation.

Finally, our system still offers a limited amount of watercolor effects compared to the traditional counterpart. Also, effects like color bleeding were simplified for performance purposes. These missing and oversimplified effects require more work which can be addressed in the future.

5. Conclusion and Future Work

We have presented a system for art-directed watercolor rendered animation based on intensity channels, which control an enhanced and extended palette of watercolor effects. Our solution performs in real-time, is production ready within Autodesk Maya and can easily be implemented into further engines. We are able to produce an extensive variety of styles, outperforming previous systems which rely heavily on parametric values for watercolor stylization. Our system manages to solve the challenge of art-direction in object-space watercolor simulations and opens new opportunities for future applications and research.

There are several ways in which the proposed watercolor simulation system can be improved and extended. The most important is to further extend the palette of simulated watercolor effects. Characteristic effects such as, for example, dry-brush (dry-on-dry technique), the high frequency turbulent noise caused by layered patches of water, the unpainted gaps at the borders of painted areas and the outlining pencil strokes found sometimes in watercolor illustrations. Effects which have already been simulated are also open for improvements. The use of control vertices can also further be optimized and extended by either adding further vertex color sets or encoding multiple effects within the color intensity channels. Finally, a watercolor system can further benefit from a dynamic paper simulation - when alternating paper textures are not desired. This feature could enhance frame-to-frame coherence and can be extended from previous work regarding dynamic canvas [CTP*03, KC05, BBT09].

Acknowledgment

We thank Davide Benvenuti, Bernhard Haux and Henriette Peter for their comments, advice and suggestions, together with the anonymous reviewers for their constructive feedback. This research is supported by the Interdisciplinary Graduate School (IGS) at the Nanyang Technological University, Singapore and the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

References

[BBT09] BÉNARD P., BOUSSEAU A., THOLLOT J.: Dynamic solid textures for real-time coherent stylization. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (2009), ACM, pp. 121–127. doi:10.1145/1507149.1507169. 7

[BCK*13] BÉNARD P., COLE F., KASS M., MORDATCH I., HEGARTY J., SENN M. S., FLEISCHER K., PESARE D., BREEDEN K.: Stylizing animation by example. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013) 32*, 4 (2013), 119. doi:10.1145/2461912.2461929. 2

[Bir87] BIRREN F.: *Principles of Color: A Review of Past Tradition and Modern Theories*. Schiffer Publishing, 1987. URL: http://books.google.com/books?id=rgeSuAAACAAJ. 5

[BKTS06] BOUSSEAU A., KAPLAN M., THOLLOT J., SILLION F. X.: Interactive watercolor rendering with temporal coherence and abstraction. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering* (2006), ACM, pp. 141–149. doi:10.1145/1124728.1124751. 2, 5, 6

[BNTS07] BOUSSEAU A., NEYRET F., THOLLOT J., SALESIN D.: Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 104. doi:10.1145/1276377.1276507. 2

[BWK05] BURGESS J., WYVILL G., KING S. A.: A system for real-time watercolour rendering. In *Computer Graphics International* (2005), IEEE, pp. 234–240. doi:10.1109/CGI.2005.1500426. 3, 6

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1997)* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 421–430. doi:10.1145/258734.258896. 2

[CT05] CHU N. S. H., TAI C.-L.: Moxi: Real-time ink dispersion in absorbent paper. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005) 24*, 3 (2005), 504–511. doi:10.1145/1073204.1073221. 2

[CTP*03] CUNZI M., THOLLOT J., PARIS S., DEBUNNE G., GASCUEL J.-D., DURAND F.: Dynamic canvas for immersive non-photorealistic walkthroughs. In *Proceedings Graphics Interface* (2003), Graphics Interface, A K Peters, LTD., pp. 121–130. URL: https://hal.inria.fr/inria-00510176. 7

[DKMI13] DIVERDI S., KRISHNASWAMY A., MĚCH R., ITO D.: Painting with polygons: A procedural watercolor engine. *Visualization and Computer Graphics, IEEE Transactions on 19*, 5 (2013), 723–735. doi:10.1109/TVCG.2012.295. 2

[FLJ*14] FIŠER J., LUKÁČ M., JAMRIŠKA O., ČADÍK M., GINGOLD Y., ASENTE P., SÝKORA D.: Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. *Computer Graphics Forum 33*, 4 (2014), 1–10. doi:10.1111/cgf.12407. 2

[Har07] HARVILL A.: *Effective Toon-Style Rendering Control Using Scalar Fields*. Memo, MAY 2007. URL: http://graphics.pixar.com/library/ToonRendering/index.html. 3

[JHN04] JOHAN H., HASHIMOTO R., NISHITA T.: Creating watercolor style images taking into account painting techniques. *The Journal of the Society for Art and Science 3*, 4 (2004), 207–215. doi:10.3756/artsci.3.207. 2

[KC05] KAPLAN M., COHEN E.: A generative model for dynamic canvas motion. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (2005), Eurographics Association, pp. 49–56. doi:10.2312/compaesth/compaesth05/049-056. 7

[LBDF13] LU J., BARNES C., DIVERDI S., FINKELSTEIN A.: Realbrush: Painting with examples of physical media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013) 32*, 4 (2013), 1–12. doi:10.1145/2461912.2461998. 2

[LC05] LEI S. I. E., CHANG C.-F.: *Real-Time Rendering of Watercolor Effects for Virtual Environments*. Springer Berlin Heidelberg, 2005, pp. 474–481. doi:10.1007/978-3-540-30543-9_60. 3, 4, 6

[LD06] LUFT T., DEUSSEN O.: Real-time watercolor for animation. *Journal of Computer Science and Technology 21*, 2 (2006), 159–165. doi:10.1007/s11390-006-0159-9. 3, 6

[LM01] LUM E. B., MA K.-L.: Non-photorealistic rendering using watercolor inspired textures and illumination. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications* (2001), pp. 322–330. doi:10.1109/PCCGA.2001.962888. 3

[Ong03] ONG K. S.: *Mastering Light & Shade in Watercolor*. International Artist, 2003. URL: http://books.google.com/books?id=1DlZNwAACAAJ. 1

[Pol13] POLLARD J. G.: *Watercolor Unleashed: New Directions for Traditional Painting Techniques*. North Light Books, 2013. URL: http://books.google.com/books?id=PDQGHOhL42gC. 1

[RD13] RIM S., DUTCHER L.: *Watercolor: Paintings by Contemporary Artists*. Chronicle Books, 2013. URL: http://books.google.com/books?id=rKJhngEACAAJ. 1

[RM00] READ P., MEYER M.-P.: *Restoration of Motion Picture Film*. Butterworth-Heinemann Series in Conservation and Museology. Elsevier Science, 2000. URL: http://books.google.com/books?id=ai_ORUt8VlwC. 1

[Sch14] SCHEINBERGER F.: *Urban Watercolor Sketching: A Guide to Drawing, Painting, and Storytelling in Color*. Watson-Guptill Publications, 2014. URL: http://books.google.com/books?id=rKJhngEACAAJ. 1

[Sma91] SMALL D.: Simulating watercolor by modeling diffusion, pigment, and paper fibers. *SPIE Proceedings 1460, Image Handling and Reproduction Systems Integration* (1991), 140–146. doi:10.1117/12.44417. 2

[VLVR05] VAN LAERHOVEN T., VAN REETH F.: Real-time simulation of watery paint. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 429–439. doi:10.1002/cav.95. 2

[WWF*14] WANG M., WANG B., FEI Y., QIAN K., WANG W., CHEN J., YONG J.-H.: Towards photo watercolorization with artistic verisimilitude. *Visualization and Computer Graphics, IEEE Transactions on 20*, 10 (2014), 1451–1460. doi:10.1109/TVCG.2014.2303984. 2, 4, 6

[YJC*13] YOU M., JANG T., CHA S., KIM J., NOH J.: Realistic paint simulation based on fluidity, diffusion, and absorption. *Computer Animation and Virtual Worlds 24*, 3-4 (2013), 297–306. doi:10.1002/cav.1500. 2