Real-time watercolor rendering of 3D objects and animation with enhanced control

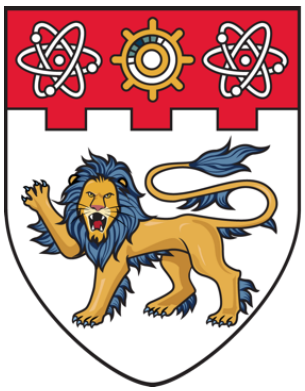
Montesdeoca, Santiago Esteban

2018

Montesdeoca, S. E. (2018). Real-time watercolor rendering of 3D objects and animation with enhanced control. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/102529

https://doi.org/10.32657/10220/47356



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

REAL-TIME WATERCOLOR RENDERING OF 3D OBJECTS AND ANIMATION WITH ENHANCED CONTROL

SANTIAGO ESTEBAN MONTESDEOCA

Interdisciplinary Graduate School
Multi-plAtform Game and Innovation Centre

2018

REAL-TIME WATERCOLOR RENDERING OF 3D OBJECTS AND ANIMATION WITH ENHANCED CONTROL

SANTIAGO ESTEBAN MONTESDEOCA

Interdisciplinary Graduate School
MAGIC - Multi-plAtform Game and Innovation Centre

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

Acknowledgements

First, I would like to acknowledge everyone whom I have had the pleasure to share quality time with. Without every single contribution that each one of you has made, I would probably not be writing this.

I would like to thank both of my supervisors, Prof. Seah Hock Soon and Assoc. Prof. Hans-Martin Rall for their support and guidance. An immense gratitude goes also to my mentor, Asst. Prof. Davide Benvenuti, for insisting on bringing me back to Singapore. Without him, I would probably be freezing in a Nordic country doing something completely unrelated to what I am doing now. I was lucky to count on the support from all three of you, motivating me and giving me the liberty to do exactly what I have always wanted to do.

My endless gratitude goes to my 'adoptive' French supervisors Joëlle Thollot, Pierre Bénard and Romain Vergne. I have looked up to you since I started this endeavour and it has been a true pleasure and an unreal learning experience working together with you. I would like to also thank my partners-in-research, Amir Semmo and Alexandre Blerón, whom I have learned a lot from.

A special thanks goes to Henriette Peter, for her invaluable company, support, encouragement and care. Without her affection, I would probably be in a very different place right now. To my family, I literally owe you my life. You have molded my personality and removed all boundaries for me to make it this far. Thank you very much for your limitless support and love.

I would also like to express my gratitude to the Singapore International Graduate Award (SINGA), the Interdisciplinary Graduate School (IGS) at the Nanyang Technological University (NTU), Singapore, the Multi-pAtform Game Innovation Centre (MAGIC) and the Maverick team at Inria Grenoble Rhône-Alpes for supporting me financially and academically during my studies.

Last but not least, I want to thank all my old and new friends in Singapore and Grenoble, without you, this PhD journey would have been quite a lonesome one.

Abstract

The research presented in this thesis pursues two goals, pushing the state-of-the art in watercolor rendering of 3D objects and animation in real-time, while at the same time providing an enhanced, interactive and art-directed control over the stylization.

Towards these goals, the field of expressive, non-photorealistic rendering is introduced, together with an introduction to the watercolor painting medium and its traditional/digital uses in animation. A literature survey follows, breaking down previous research and approaches towards a watercolor synthesis.

The contribution begins addressing existing limitations by introducing a custom painterly 3D material that brings colors to the foreground of shading control and enables dilution of colors and cangiante illumination. More importantly, this material embeds the locally art-directed and procedural stylization parameters that drive the future watercolor effects in a spatially coherent manner.

The watercolor stylization is then addressed, by extending and improving upon the palette of watercolor effects, introducing different methods and algorithms to synthesize pigment-, edge-, substrate- and abstraction-based watercolor effects. These include novel approaches towards art-directed color bleeding, dry-brush, gaps & overlaps and dilution in object-space. Together with enhanced and optimized approaches to produce controllable and robust gradient edge darkening, depth aware substrate distortion, substrate granulation and deferred substrate lighting.

The contribution is then complemented by presenting the real-time art-direction throughout the interaction spectrum, which is generalized through style control semantics that allow art-directed results to be visualized in different styles within a direct stylization pipeline. This endeavour was implemented in a widely used digital content creation software, which enabled the opportunity to perform two different user-studies involving professional computer graphics artists. The user studies substantiated and validated the contribution of this work, which can be applied in many professional fields that require art-directed 3D graphics.

Publications

The work presented in this thesis appeared in the following publications:

Montesdeoca S. E., Seah H. S., and Rall H.-M. 2016a. Art-Directed Watercolor Rendered Animation. In *Expressive '16: The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering (NPAR '16)*. The Eurographics Association, 51–58. DOI:https://doi.org/10.2312/exp.20161063

Montesdeoca S. E., Seah H. S., and Benvenuti D. 2016b. Artistically Driven Non-Photorealistic Animation. In *SAS2016 28th Annual Conference of the Society for Animation Studies*. The Society of Animation Studies.

Montesdeoca S. E., Seah H. S., Rall H.-M., and Benvenuti D. 2017a. Art-Directed Watercolor Stylization of 3D Animations in Real-Time. *Computers & Graphics* 65 (2017), 60–72. DOI:https://doi.org/10.1016/j.cag.2017.03.002

Montesdeoca S. E., Seah H. S., Benvenuti D., Bénard P., Rall H.-M., Thollot J., and Vergne R. 2017b. Direct 3D Stylization Pipelines. In *ACM SIGGRAPH 2017 Real Time Live! (SIGGRAPH '17)*. ACM, 18. DOI:https://doi.org/10.1145/3098333.3098339

Montesdeoca S. E., Seah H. S., Bénard P., Vergne R., Thollot J., Rall H.-M., and Benvenuti D. 2017c. Edge- and Substrate-Based Effects for Watercolor Stylization. In *Expressive '17: The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering (NPAR '17)*. ACM, Article 2, 10 pages. DOI:https://doi.org/10.1145/3092919.3092928

Montesdeoca S. E., Seah H. S., Semmo A., Bénard P., Vergne R., Thollot J., and Benvenuti D. 2018. MNPR: A Framework for Real-Time Expressive Non-Photorealistic Rendering of 3D Computer Graphics. In *Expressive '18: The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering (NPAR '18)*. ACM, Article 9, 11 pages. DOI:https://doi.org/10.1145/3229147.3229162

Chiew Y. X., Seah H. S., and Montesdeoca S. E. 2018. Real-Time Art-Directed Charcoal Cyber Arts. In *2018 International Conference on Cyberworlds (CW)*. IEEE.

Contents

List of Figures

List of Tables

1. Introduction

The beauty of skillfully painted traditional watercolor artworks is irrefutable (Figure 1.1). Having gained its place in art and an increasing popularity since the second half of the 18th century [Hargraves et al., 2007; Shanes et al., 2000], watercolors have been a natural painting medium of experimentation. From the unforgiving transparent nature to the capricious fluidity that enables its characteristic effects, traditional watercolor presents itself as a unique challenge—even for the experienced artist [Curtis and Capon, 2005; Edin and Jepsen, 2010, p. 11, 19].

When watercolors are painted over a series of papers in animated form, the challenge for the artists grows exponentially, due to the volatile qualities of the natural medium that will not allow two paintings to ever be the same [Cartwright, 2015, p. 61]. This aspect makes the use of traditional watercolors in animation almost unfeasible for production (Section 2.2). Therefore, a computer aided solution would be ideal to generate watercolor aesthetic in animation.

The body of work presented in this thesis addresses the application of watercolors in animation, offering an alternative synthetic medium to render images from animated 3D elements in a watercolor style. This objective presents many research problems in computer graphics that need to be studied to achieve: (1) the natural look of watercolors in 3D objects and animation; (2) an interactive and artist-friendly tool-set that offers enhanced control to maximize the art-direction of the watercolor look. These studies fall into the field of *Non-Photorealistic Rendering*, often referred to as *Expressive Rendering*, which is introduced next.

Looking into the Sky, © Richard Chao

The Tram, © Orhan Gürel

Figure 1.1: Traditional watercolor paintings by renowned artists.

1.1 Non-Photorealistic Rendering

Non-photorealistic rendering, commonly abbreviated by its acronym NPR, is a broad field in computer graphics. It concentrates on computer generated imagery (CGI) that strives against a realistic recreation, towards a stylized and expressive depiction. Many researchers concur that the term non-photorealistic by itself is fuzzy [Durand, 2002] and that the notion of realism can differ depending on context [Ferwerda, 2003]. Nonetheless, the term "Non-Photorealistic Rendering" has been ingrained in the computer graphics field.

NPR has been studied since the emergence of computer graphics—either directly, or indirectly as a side-product in the pursuit of photorealism. Most of the dedicated contributions to the field have been presented at a specialized conference called *Non-Photorealistic Animation and Rendering (NPAR)* since 2000—which has fused into the *Expressive joint Symposium* since 2012. As such, NPR presents a significant gamut of research, introducing models, algorithms and systems, which have found practical applications in fields that rely on computer graphics such as print, entertainment and visualization [Winnemöller, 2013] (Figure 1.2). However, the application of non-photorealistic rendering in animated 3D computer graphics (e.g., animated films, games) has been limited, especially when compared to photorealistic rendering.

Print: amplitude modulation halftoning, Martín et al. [2017]

Entertainment: oil painting VFX, © Interscope Communications - What Dreams May Come (1998)

Visualization: markers, © Palette CAD GmbH - PaletteCAD

Figure 1.2: Examples of different applications of NPR techniques.

NPR in animated 3D computer graphics

With the plethora of animated 3D content coming out each year, it has become increasingly critical for productions to differentiate themselves from others. The animation and games industries are producing more content than ever, as new players worldwide come into an already saturated market [Hopewell, 2018; Parrish, 2018; Verrier, 2013]. This abundance has diluted the audience and repercussions have

started to become noticeable even at big established studios in both industries like *PDI Dreamworks Animation*, *Crytek* and *THQ* [Fritz, 2013; Good, 2016; Verrier, 2015]. While it is true that there are many factors involved in layoffs and closures at studios, one thing is clear: these industries have matured to the point where new content is not enough, it has to stand out from the crowd.

Most 3D content today, especially in the animation industry, is dominated by photorealistic rendering approaches. This has led to a limited variety of looks and a growing list of animated films that possess a generic look—often referred to as *the CG look*—due to photorealism becoming the standardized computer graphics (CG) aesthetic (Figure 1.3). Even many *AAA*[1] games are continuously pushing their renders towards photorealism and, as real-time technologies improve, keep aiming towards this aesthetic. Since its inception, most of the computer graphics field has been on a quest towards photorealism [Durand, 2002; Salesin, 2002], which is reflected in the tools and software that exist nowadays. Therefore, the potential of different NPR styles for 3D animated productions has often been overlooked, creating a rather untapped area within the computer graphics field for production ready NPR.

© Disney-Pixar - Coco (2017) © Illumination Mac Guff - Despicable Me 3 (2017)

Figure 1.3: Examples of recent animated films presenting *the CG look*.

There is one main exception of an NPR style that has been widely incorporated in animated productions and games alike, anime. Anime productions, being traditionally hand drawn, have been incorporating 3D computer graphics into their production tool-set since the 1980s [Spencer, 2017]. While initially used only for certain animated props, environments or creatures—that would have been too time consuming to draw—, anime NPR is now being successfully used to render entire animated productions [Kelts, 2016]. This was made possible thanks to the high demand and the wide research that has concentrated on this style (i.e., Lake et al. [2000]; Todo et al. [2007, 2013]), leading to dedicated software like *Pencil+ 4* and *Maneki* that are commercially available and specialize in rendering the anime aesthetic.

[1] *AAA* is an informal classification used for video games developed by a large studio, typically with higher development and marketing budgets. The term is analogous to the film industry term "blockbuster"[Schultz, 2018].

Apart from anime, some established animation studios are beginning to recognize the potential of NPR and have been experimenting with different styles on their 3D animated short films. Two excellent examples came from the *Walt Disney Animation Studios* in the form of *Paperman* (2012, Figure 1.4a) and *Feast* (2014). These shorts have been successful with the audience and critics alike, each earning an *Academy Award for Best Animated Short Film* (*Oscar*®). However, the different technologies that helped craft the NPR look of these short films are not commercially available. This unfortunate situation, together with the additional *R&D* (research and development) and overhead in artists' work, is severely limiting the adoption of different NPR styles in animated productions by individual artists and small studios—who cannot afford to reproduce these looks. Nonetheless, as studios keep pushing towards differentiating their animated content from each other, new NPR productions are starting to surface e.g., *One Small Step* (2018, Figure 1.4b) by *Taiko Animation Studios*.

© Disney - Paperman (2012) © Taiko Animation Studios - One Small Step (2018)

Figure 1.4: Examples of animated NPR productions.

I argue that NPR approaches are a promising alternative for the creation of CGI that will allow unique productions that differentiate themselves from each other, and will be of increasing relevance in the years to come. This argument is best elaborated, when we compare photorealistic and non-photorealistic rendering metaphorically with their representations in analogous media within art. From this perspective, photorealistic rendering is comparable to photography and photorealistic paintings. As such, it is mostly based on the reality of the captured subject and bound by the rules of physics. In contrast, non-photorealistic rendering is comparable to paintings in any style. As such, it is based on the artist's individual expressive manifestation of the subject and unbound by rules.

In visual arts, photography and photorealistic paintings represent a rather narrow spectrum. In comparison, stylized paintings have been subject to individual representations and art-movements throughout history, contributing significant variety in the visual arts and representing a much wider spectrum altogether. Therefore, considering photorealistic and non-photorealistic rendering, the 3D synthetic representatives of their comparable analogous media in art and art history, we can confidently expect NPR to gain a major role within 3D content. However, in the

same way that scientists concocted new colors and artisans crafted new tools for painters to enable their visions, we need to provide these to 3D artists. This is where the area of *Expressive Rendering* comes into the spotlight.

Expressive Rendering

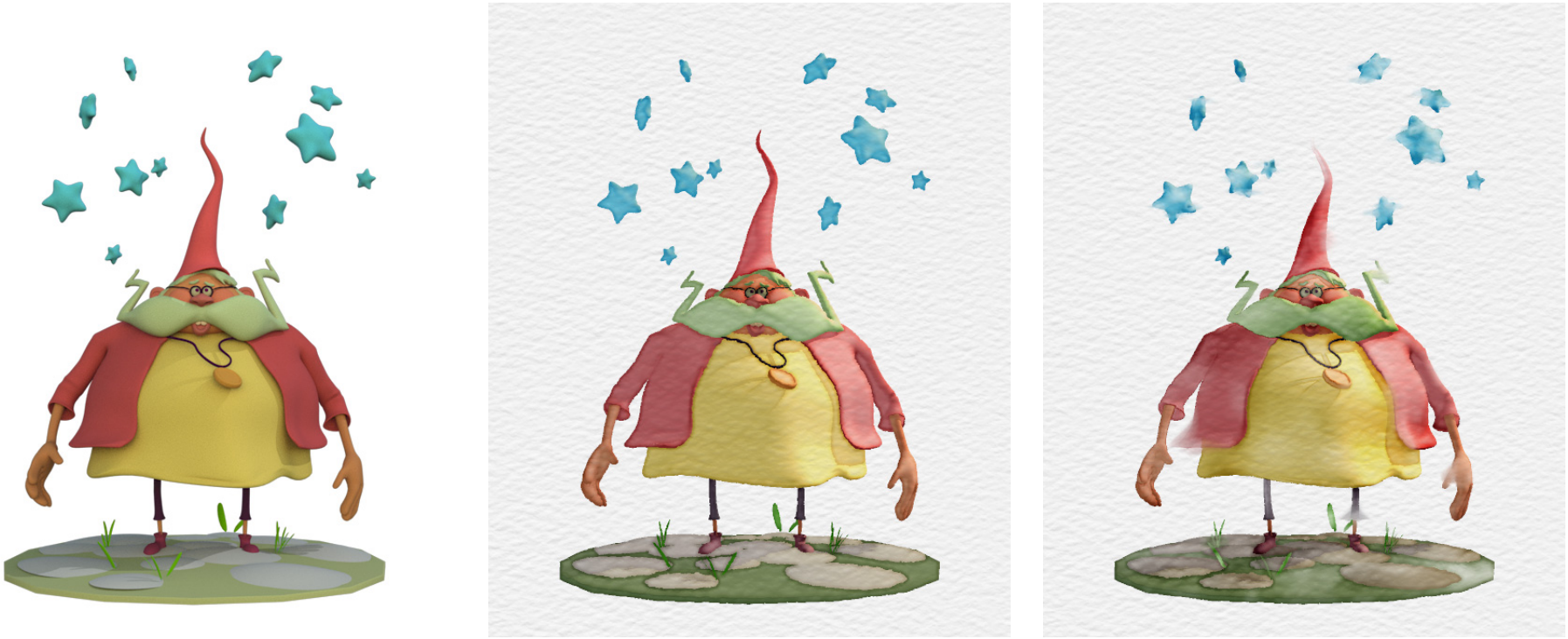
Expressive Rendering is often used interchangeably with Non-Photorealistic rendering. However, in the context of this thesis, I would like to make a clear distinction between them. Expressive rendering refers to rendering with the ability to art-direct the rendered imagery that is produced from 3D data, enabling an individual vision in a specific style. After all, 'the artistic medium alone is certainly not enough if the rendering approach shall reproduce the artistic process as a whole' [Luft et al., 2008][p. 60].

Rendering, be it photorealistic or non-photorealistic, is not expressive by itself. Photorealistic rendering has pushed towards expressiveness since its inception, offering tools to let artists control the lighting in the scene, the material attributes [Schmidt et al., 2016] and tweak the overall photoreal look at the compositing stage. These tools have shown considerable success, to the extent where skilled artists often achieve expressive results that simulate a custom reality, indistinguishable from a photograph.

Non-photorealistic rendering also takes advantage of photorealistic expressive tools and provide a further alternative by releasing them from any physical laws. Additionally, depending on the style of the render, special effects are included that help emulate a specific look. Correspondingly, these effects require custom tools at different levels of control—depending on the intended audience and use. There are many NPR approaches in 3D computer graphics (i.e., filter-based, example-based, mark-based) and each approach might require a different tool-set.

Expressive NPR is a complex and extensive field which not only deals with the aesthetic of each style, but also its interactivity and art-direction—which can be completely different depending on the NPR approach. I concentrated my efforts on investigating and developing expressive rendering to generate a stylized watercolor outcome for 3D objects and animation in real-time (Figure 1.5).

> *Overall, NPR researchers might put more emphasis on assisting art creation, rather than automating it.* (Holger Winnemöller [2013])



Photorealistic render (Arnold)

Non-photorealistic render

Expressive non-photorealistic render

Figure 1.5: Comparison of rendering methods. Photorealistic renders follow physical rules constrained to reality, featuring a *CG look*. Non-photorealistic renders do not need to follow physical rules, presenting different reflectance models, effects and marks, which are not based in reality. Expressive non-photorealistic renders allow the art-direction of non-photorealistic attributes to fit a specific artistic vision. *Wiz Biz* model, CC BY Tom Robinson.

1.2 Motivation and Research Problem

The research conducted during my PhD studies was motivated from my artistic need to generate non-photorealistic looks using 3D computer graphics. Since my first 3D animated short film in 2010, I strove to achieve unconventional looks that would come closer to the painted artwork that inspired the film aesthetic. Unfortunately, there are scarce and limited tools available for artists to achieve non-photorealistic 3D styles that look as if painted by hand.

The currently practised workflow is to use painted textures mapped onto 3D objects, render them in a semi-photorealistic manner and later tweak the resulting images at the composting stage to achieve the final stylized look. However, this three-stage pipeline is rather disconnecting from the creative process and counter-intuitive for artists. They are only able to see the outcome at the final compositing stage, which means that artists find themselves constantly trying to predict how the aesthetic will develop at later stages. This is even worse when each of these stages is handled by different artists and software—which is usually the case. Additionally, if the end result is not right, any changes require artists to go up the pipeline, make the change (trying to predict the outcome), re-render and composite the stylized look back together. The inability to see the stylized outcome is arguably one of the biggest creative bottlenecks for non-photorealistic 3D productions. What if this workflow could be dramatically optimized to work in real-time?

I have always been fascinated by traditional watercolor paintings and 3D animation. Thus, researching how to synthesize watercolor into 3D animation is something I have always wanted to do, but I never had the opportunity, the time, nor the mentorship to fully dedicate myself to it. My goal during my studies was to make a watercolor look possible and accessible for 3D productions, by developing new artistic tools to portray stylized 3D computer graphics, which can be easily adopted by current production pipelines and expanded into other styles. The overall contribution of my work aims to bring variety to otherwise generic looking 3D productions, and make traditional looking animation possible using modern technologies and artist-oriented tools.

Watercolor is a highly art-directed traditional medium, whose qualities bring exciting opportunities to further research in expressive non-photorealistic rendering. With the many characteristic visual effects of watercolors that are thoughtfully used and placed by artists in a painting, there is much to address in terms of (1) synthesizing the natural look and (2) allowing extensive art-direction of the medium. Towards these goals, various approaches have been proposed to synthesize watercolors, which are meticulously presented in the literature review of related work (Chapter 3).

Substantial research in non-photorealistic rendering concentrates on synthesizing watercolor-looking imagery through brush strokes or images/photographs. However, little has been done to synthesize 3D animations into animated watercolors in real-time. Synthesizing a watercolor look from 3D animation brings a specific set of problems and research questions, which are addressed throughout this thesis.

1. How can 3D data be transformed into watercolor-looking images?
2. How can characteristic effects of watercolors be reproduced?
3. How can local effects remain spatially coherent during animation?
4. How can the synthesis of watercolor effects be art-directed?
5. How can all this work in real-time?

I have spent the past 4 years of my PhD studies focusing on these research questions and proposing an optimized and user-friendly way to synthesize 3D watercolor animations in real-time.

1.3 Contributions

The work performed during my PhD candidature at the Interdisciplinary Graduate School of the Nanyang Technological University in Singapore, presented in this thesis, has produced multiple contributions in the pursuit of real-time watercolor rendering of 3D objects and animation with enhanced control. These contributions are split into three chapters, each with a functional role towards the intended interdisciplinary research objective.

Figure 1.6: Our synthesized result. *Summer Afternoon* model, © Stevie Brown.

Painterly materials. A real-time painterly material for 3D objects is introduced (Chapter 4), which proposes an alternate way on how light is reflected from objects, bringing colors into the foreground of shading control. This custom material is intended to replicate the watercolor painting thought process, but still support texture mapping techniques and reflectance models from common real-time materials. The methods described herein customize the diffusion of the reflectance model—with enhanced control over the shade and light areas within objects—that support the dilution of color and cangiante illumination. More importantly, the material is the effect driver, rendering the embedded object- and material-space parameters to control the watercolor stylization in the stylization pipeline. Apart from defining the rendered color, and driving the effects, the material outputs a series of additional object-space data into multiple render targets to help the stylization process.

Watercolor stylization. The watercolor stylization takes place at various stages, emulating different characteristic effects that help produce the watercolor synthesis (Chapter 5). The synthesis is done through image processing, using several 2D images that contain 3D data rendered by the painterly materials. The stylization concentrates on emulating characteristic pigment-, edge-, substrate- and abstraction-based effects that, when used in unison, can reproduce a watercolor

render (Figure 1.6). The proposed methods and algorithms of each stage allow the synthesis of characteristic effects such as: pigment turbulence, dry-brush and granulation, edge darkening, gaps & overlaps, substrate distortion and color bleeding. Each effect, while synthesized differently, is controlled in real-time through the direct stylization pipeline.

Direct stylization pipeline. The direct stylization pipeline is the framework that enables expressive control over the different watercolor effects and proposes style control semantics to allow cross-stylization into other media (Chapter 6). The tools to art-direct the stylization effects proposed herein were designed and developed to cover the interaction spectrum [Isenberg, 2016]. From a high level of control using style presets and global control parameters, to a medium level of control using material presets and material control parameters, down to a low-level of control with mapped control parameters. These levels are complemented by proxy control, which decouples the parameters from the subject by using invisible control proxies. All the art-direction tools drive the characteristic effects of the watercolor stylization in real-time by following a control scheme, governed by style control semantics. These control semantics are also proposed, which allow the art-direction to be used with different stylizations that adhere to the same control scheme. The cross-stylization is described and demonstrated by transferring the art-direction to other non-photorealistic styles, such as oil painting and charcoal drawing. The direct stylization pipeline was built on top of Autodesk® Maya®, a popular digital content creation (DCC) package. While this guaranteed a wide user-base, the choice of a closed-source framework brought many caveats. The contribution of this thesis concludes with a thorough discussion on the implementation of the stylization framework within this software.

The remainder of this thesis is composed of a background chapter (Chapter 2), meant to (1) bring the reader to a fundamental understanding of watercolors as a traditional medium (Section 2.1); (2) describe how watercolors have been previously used in animation (Section 2.2); and (3) expose how watercolors are nowadays employed digitally (Section 2.3).

The literature review of related work (Chapter 3) is introduced, which surveys the most relevant research done in stroke-, image-, object- and hybrid-space watercolor techniques. This review is complemented by a survey on existing frameworks for expressive rendering in object-space.

Finally, this thesis concludes (Chapter 7) with a summary of contributions and the proposed venues for future work in the field of expressive watercolor rendering. Further details on the watercolor stylization pipeline, together with the raw data of the conducted user studies are also available in Appendix A.

2. Synthesizing Watercolor

To do a proper synthesis of watercolor, we need to understand the traditional medium, survey what has been previously done with traditional watercolors in animation and what is possible to do nowadays in the digital realm. This chapter covers this background to bring all readers to the same level of knowledge prior to diving into the technicalities of the related work and the contribution of the presented research.

First, a compact overview of traditional watercolors is given (Section 2.1). The components of traditional watercolors are presented, together with important terminology that will be used in the remaining chapters. Then, the characteristic effects, commonly found in watercolor are categorized and introduced. These effects are fundamental to reproduce a watercolor aesthetic and will often be referred to in later chapters.

With the gained knowledge of watercolors as a traditional medium, a brief, but holistic study of traditional watercolor in animation throughout history is presented, spanning the last 100 years (Section 2.2). This chapter then concludes by surveying the medium's digital counterpart with the latest commercial technologies that enable computer generated watercolor (Section 2.3).

Figure 2.1: An artist painting with watercolors. Photo, © Nik MacMillan.

2.1 Traditional Watercolors

Traditional watercolor, in its most fundamental arrangement, was probably the first form of paint ever used because its core elements, water and color pigments, are found anywhere and are highly prone to mix. But, as easy as it can be to paint in its elemental form, it has a fragile nature which makes it highly sensitive to touch, light, humidity and additional water. Little evidence exists as to how many years back its origins date, however, it is said that it has its rough roots probably since the cave paintings of *Paleolithic Europe* [Dutcher, 2013]. Back then, depictions of bison and other beasts where painted with 'water reds, ochers from the soil and black from burnt wood' [Gwynn, 2004]. However, watercolors in the western world, as we know them, appeared during the *Renaissance* in the 15th century and is, nowadays, one of the main traditional painting media used by amateurs and professionals alike.

To better approach the simulation of watercolors digitally, it is crucial to understand how traditional watercolor works. There is extensive literature available about watercolor painting. Most of it focuses on exposing watercolor artwork and the biography of specific artists, some explain the history of watercolor painting and its influential painters and there are numerous examples which also explain how to approach an actual watercolor painting—with its required workflow. They all have their distinctive target audiences, from art enthusiasts to art historians and watercolor aficionados/artists. Nevertheless, few technical details are given as on why watercolor behaves the way it does. Some try to offer some insight on what is happening behind the guidelines used to achieve a certain effect, but none really goes too much into detail—it presents itself as a real challenge due to the scarcity of available research on the physical processes involved [Curtis et al., 1997].

This section will focus on summarizing the literature and filling potential gaps involving both, the aesthetic intricacies and the physical background, for the digital implementation of watercolors. The aim is to demonstrate what the technology presented in this thesis strives to accomplish. To achieve this, traditional watercolors are going to be deconstructed. We begin by identifying the components of watercolors and their distinctive elements that give traditional watercolor its characteristic qualities. From the ingredients of watercolor paint and its commercial properties, to the different types of watercolor paper and tools of trade of watercolor painters. The characteristics of watercolors will then be explored and elaborated upon. From its distinctive properties like transparency and surface dependency to the characteristic effects which these properties generate. In a computer generated synthesis, only the final rendered look matters. However, to achieve the final look, a lot of references to the traditional techniques are made and used to portray different effects.

2.1.1 Components

For simplicity and classification, the components that give traditional watercolor paintings its characteristic qualities are three: watercolor paint, paper and tools. The interplay between these three components can bring up different physical behaviors, that result in unique aesthetic effects.

The three components will be introduced and elaborated upon, as each can drastically influence the final painting. This aspect makes controlling traditional watercolors a challenging endeavor with variables that significantly affect the visual outcome.

Watercolor paint

The main component of a traditional watercolor painting is the watercolor paint. But, as simple as it sounds, there is much more to watercolor paint than the elements it is named after. Color pigments are highly prone to mix with water, but as soon as the water dries out, pigments return to their granular state—making them again highly susceptible and vulnerable to any external influence. For color to adhere to surfaces, a sticky material is needed. This key element is the one that differentiates all different types of paint—whether it is acrylic, oil or watercolor paint [Saitzyk, 1987].

In the case of watercolors, paint still follows a century-old recipe that has remained unchanged and is available in many forms e.g., dry cakes, semi-moist pans and tubes. The ingredients are the following:

- *Dying agent.* Small particles of color, referred as pigment, from natural source materials like plants, minerals, metals, oxides or man-made synthesized sources (e.g., cobalt salts of alumina, rose madder, alizarin).
- *Colloid agent.* Sticky binding material which holds color together and helps the pigment adhere to the painting surface. In the specific case of watercolors, this agent is called Gum Arabic—which is harvested from the Acacia Seyal and Acacia Senegal originated from Sudan [Scheinberger, 2014].
- *Additives.* Additional materials which alter the appearance and malleability of the paint (e.g., glycerin, ox-gall, honey).
- *Solvent.* Liquid which serves as a vehicle, it dilutes and disperses the pigments. In the specific case of watercolors, water.

Figure 2.2: Watercolor in the form of dry cakes. Photo, © Denise Johnson.

Even though this recipe is also used to create gouache paint, purists consider gouache a corruption of the watercolor practice [Smith and Ardill, 2011]. By adding white pigment, the characteristic transparency of watercolors is lost, creating an opaque medium. Due to this change of property, gouache is considered a paint medium on its own. Watercolor paints still vary depending on the pigments' source material and paint manufacturing. The basic defining properties and their values found in commercial watercolors nowadays are the following:

- *Transparency.* Defines the translucency of the paint. Transparent colors flow much easier than opaque [Abgott, 2007].

- *Granulation.* Defines the density of pigments. These granulating or sedimentary colors tend to settle into the valleys of the paper creating additional texture. Non-granulated paint flows much easier than granulated.

- *Staining.* Defines the staining power. Non-staining colors can be lifted out if the surface is re-wet. Staining colors soak into the paper, staining it.

- *Permanence.* Defines the amount of time before the pigment fades. It is usually linked with the quality of the pigment and its price.

When mixing colors, the properties of the individual pigments are kept. This contributes to unforeseen side effects which have to be taken into account when painting traditionally. Mixing a fugitive color with a permanent one will eventually change the color over time. Additionally, mixing a granular color with a non-granular color will result in a higher concentration of the granular pigment on the valleys of the paper, causing a color separation. There are several other behaviors which happen when mixing specific pigments, nevertheless, none of them are crucial for the implementation of watercolor rendering of 3D scenes and are out of the scope of this thesis.

Watercolor paper

Another equally important part of a watercolor painting is the substrate (surface) it is painted on. Watercolor paper and its profile structure have a significant influence on how the watercolor paint behaves, dictating the range of expression and determining the character of the work [Worth and Deutch, 1982].

Traditionally, watercolor has been painted on paper made from interlaced fibers of cotton, and it is still the case nowadays. Cotton happens to be acid free and naturally white which provides a long lasting, portable surface which is free of chemicals. An additional coating/sizing of gelatin, starch or *Aquapel* is also given to the paper. These guarantee an evenly spread of color and optimum amount of absorption, without much loss of luminosity.

Watercolor papers are traditionally handmade, but the process has been heavily industrialized nowadays. There are mainly three different types of watercolor paper available, each one with its specific structure and characteristic.

- *Rough.* As the name implies, rough paper has the roughest surface industrially available, which causes wet paint to precipitate into the valleys of its surface. This gives texture, but causes irregular edges which make detailed painting challenging. Additionally, if the paint is not wet enough, the paper crevices remain unfilled, letting the white valleys of the paper show through (dry-brush effect) [Gwynn, 2004].
- *Cold-pressed.* Also known as *Not* (not hot-pressed), is a pressed rough paper. It possesses a toned down rough surface which retains some characteristics of the rough paper. Nonetheless, it is an intermediate paper which is still suited for large smooth washes and details.
- *Hot-pressed.* This paper is a rough paper which has been pressed at a high temperature. Because of this, it presents a smooth surface, which is most suitable for drawing and painting in fine detail.

Figure 2.3: Arches Aquarelle papers. Scans, © Bruce MacEvoy.

Each type of paper also comes in different thicknesses, which enable increasing abuse and amounts of water, without showing signs of paper damage and cockling/buckling. Additionally, more water enables the paper to remain wet for longer which benefits wet-in-wet techniques e.g., pigment turbulence, color-bleeding.

Watercolor painting tools

The most essential tool for watercolor paintings is the brush, which is normally used to apply the paint onto the paper. However, there are numerous additional tools which can be used to achieve certain effects more easily and even enhance a watercolor painting. These tools will not be inspected in detail, as the work in this thesis is concerned with the final watercolor look, not with the specific processes involved in painting watercolors—which are more relevant for stroke-space approaches (Section 3.1). Nonetheless, the most common tools will be briefly mentioned, as these are still relevant for the traditional medium and present unique behaviors, which can be materials for future research.

- *Brushes.* These can be of various sizes and their hair can be from different sources e.g., kolinsky, ox, squirrel, camel and synthetic. Brushes are usually categorized into three types: (1) the round brush, which harvests the most water and paint, (2) the flat brush, which contains less water and a different hair arrangement and (3) scrubbers which, help lifting wet pigment out and softening edges [Abgott, 2007].

- *Masking fluid.* A liquid latex solution which can be applied directly to the paper to mask certain areas and preserve the substrate's original color (mostly white). The solution can be later removed to paint over the masked area or leave, as is. This technique is called "stopping out" and can be achieved with adhesive tape, as well.

- *Ox gall.* An organic liquid solution, which augments the dispersion of watercolor paint over the substrate.

- *Salt.* Mineral composed primarily of sodium chloride (kosher salt), which absorbs wet watercolor paint, creating fractal looking shapes.

- *Sponge, rags, tissue.* Soft tools used to lift out wet color from a painting or paint with a specific texture.

- *Eraser, knife and sandpaper.* Hard tools used for scraping/rubbing away paint from the substrate.

- *Toothbrush.* Tool used to speckle watercolor paint into the painting.

All these different components give watercolors its characteristic properties and, used in conjunction, its characteristic effects, presented next.

2.1.2 Properties and Characteristic Effects

Watercolor started as simple colored drawings and evolved to a medium used for any type of representations in full-fledged paintings, rivaling the uses of oil painting and sometimes even emulating it. Contemporary watercolor uses are still present in "high art", but a renewed interest has arisen for painted sketches due to the immediacy and high versatility of the medium. Watercolors are widely used nowadays and, while their uses vary immensely, there are certain properties—which watercolor has over other natural media—that give the medium its unique look. The main properties identified are the following:

- *Transparency.* When applied thinly, most watercolor pigments allow light to go through and reflect anything underneath, be it the paper itself or another painted layer. Artists take advantage of this property by mixing colors optically in layers, painting translucent or bright areas with slight glazes and exploiting the power of negative painting.
- *Governed by gravity.* Watercolor is susceptible to multiple effects caused by gravity and the way the solvent/vehicle (water) reacts to it. Artists take advantage of this property by regulating the amount of water and using pigments with different densities.
- *Prone to disperse.* Paint is highly prone to flow wherever the vehicle/solvent (water) is still wet. This high volatility presents a challenge to the artist to fully control watercolor paint. Notwithstanding, this property can be exploited by the artist to achieve additional effects i.e., pigment turbulence and color bleeding.
- *Substrate dependent.* Watercolor is usually painted on cotton paper, which has an inherent profile structure—if not hot-pressed. This presents additional textural effects that other media, painted on other substrates, do not possess. Depending on the desired effect, different substrates can be chosen for the painting.

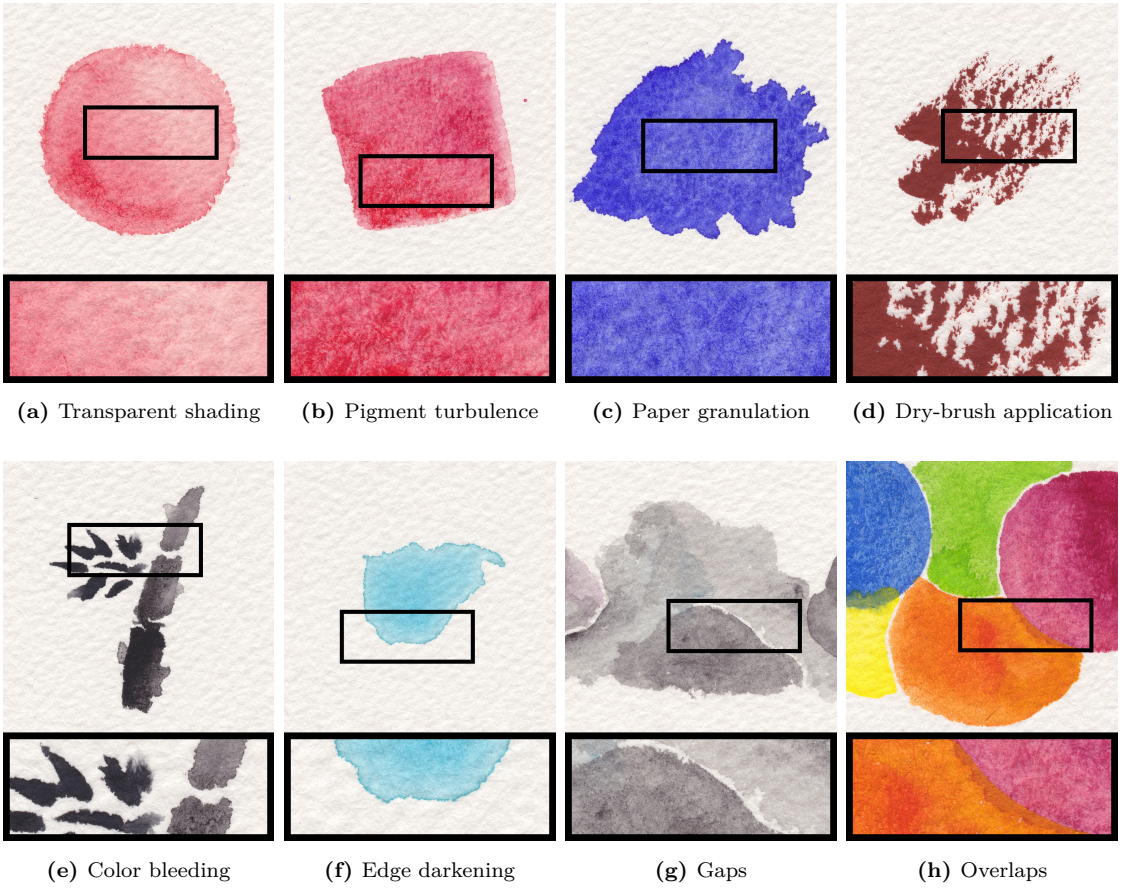
All these properties together are responsible for various characteristic effects of watercolor. While not all of them are obviously present in every watercolor painting, these effects need to be reproduced when approaching a credible digital synthesis of the medium. Curtis et al. [1997], the first researchers specifically addressing the digital synthesis of watercolors, remarked the importance of recreating these effects and proposed the physically-based synthesis of an initial set of six effects. Nonetheless, considering that we are not simulating the painting process, but rather the final synthesized look of watercolors, the characteristic effects can

be re-organized and extended for our watercolor synthesis, based on our own observations and analysis from the extensive literature on watercolor painting [Abgott, 2007; Brooks, 1977; Dutcher, 2013; Gwynn, 2004; Hargraves et al., 2007; Hoffmann, 2012; Ong, 2003; Pollard, 2013; Scheinberger, 2014; Shanes et al., 2000; Smith and Ardill, 2011; Taylor, 2003; Tedeschi et al., 2008; Worth and Deutch, 1982]. From the vast literature, a series of recurring visual effects with unique aesthetic characteristics was extracted, which we consider essential for the synthesis of a digital watercolor aesthetic (Figure 2.4).

- *Transparent shading.* The use of transparency and color to depict form, instead of lightness values. This effect often generates alternative reflectance models that deviate from photorealism (Figure 2.4a).
- *Pigment turbulence.* A low-frequency pigment noise, creating areas of varying color density over an otherwise flat colored area (Figure 2.4b).
- *Paper granulation.* When pigments settle in the valleys of the paper, presenting a denser pigmentation (darker) compared to the peaks of the paper (Figure 2.4c).
- *Dry-brush.* When pigments settle only on the peaks of the paper, leaving the valleys untouched (Figure 2.4d).
- *Paper distortion.* When the roughness of the paper profile slightly modifies the placement of pigments, creating small distortions in the subject's depiction (Figure 2.4).
- *Color bleeding.* When colors flow (bleed) within a wet area to blend with other colors or with the substrate (Figure 2.4e).
- *Edge darkening.* When pigments accumulate at the borders of wet patches of water, gradually concentrating pigments towards the edge with a darker color (Figure 2.4f).
- *Gaps & Overlaps.* When the transition between nearby edges presents either a gap or an overlap, generating a sketchier look (Figures 2.4g and 2.4h).

While we concentrate on the visual qualities of these effects for now, the physical phenomena that produce these visual effects are explained later in Chapters 4 and 5. The coexistence of these effects is what gives watercolors its distinctive look over other natural painting media. Therefore, synthesizing them is the key to generate digital watercolor images.

Naturally, there are more characteristic visual effects that are of relevance for a truthful watercolor synthesis (e.g., compound pigmentation, pigment clumping),



(a) Transparent shading **(b)** Pigment turbulence **(c)** Paper granulation **(d)** Dry-brush application

(e) Color bleeding **(f)** Edge darkening **(g)** Gaps **(h)** Overlaps

Figure 2.4: Essential characteristic effects of watercolor.

but these have been excluded from our studies due to the limited time and the additional focus on the art-direction of these effects. Nonetheless, the remaining visual effects present exciting venues for future research in the pursuit towards a faithful watercolor synthesis.

Apart from the characteristic effects of watercolor, paintings in general possess extensive expressive qualities, which are individual to each artist who translates his/her view into the canvas. Therefore, the work performed in this thesis also focuses on enhancing control to be able to art-direct the visual effects in the pursuit for an expressive non-photorealistic solution.

Now that watercolor as a traditional painting medium has been introduced, with its components, properties and characteristic effects, it is time to present its application in motion in the form of animation.

2.2 Watercolors in Animation

Watercolor films are a rare sight among the vast range of traditionally animated films. Therefore, it comes to no surprise that this technique has been overlooked by most filmmakers and the audience alike. Animation usually requires at least 12 drawings per second in order for the audience to have a satisfactory motion perception [Read and Meyer, 2000, p. 11, 24]—which is often referred to as "animating in twos" in a 24 frames per second film. Therefore, creating consecutive paintings to create an illusion of life in animation, with watercolors, would involve an immense amount of time, talent and budget—making the traditional medium impracticable for animation. However, against all odds, watercolor is still being used in animation.

This section brings a brief survey of traditional watercolor in animation, and its diverse global history which spans close to 100 years. From the origin of color in film cartoons, which featured watercolor backgrounds made at *The Walt Disney Company*, to the ink-wash animation techniques used by the *Shanghai Animation Film Studio*. Special attention is also given to contemporary watercolor animated productions, which often take advantage of digital tools to open new possibilities.

2.2.1 The Early Days

The role of watercolor in films started immediately after *Technicolor* technology enabled cartoons to have colors for the first time in 1932. It was *Flowers and Trees* (1932), a *Disney* short that was part of the *Silly Symphonies*, where watercolor initiated its role in animation [Barbagallo, 2003]. Nevertheless, watercolor was exclusively used for the background art and it rarely made it out of its mere background role—at least at *Disney*.

Watercolor had been widely used before in children book's illustrations so it ended up being a natural step to follow—with the new possibility to produce color animations. Renowned children books illustrator, Gustav Tenggren, joined *Disney* and helped with the development in this new style [Bacher, 2007]. Further experiments followed with the medium and a series of short films in the *Silly Symphonies* followed like *Elmer Elephant* (1936) and *Three Little Wolves* (1936).

The experimentation led to the first animated feature film with color and sound in history, *Snow White and the Seven Dwarfs* (1937). The film featured gorgeous

Figure 2.5: Elmer Elephant (1936), © Disney.

watercolor backgrounds together with animated characters drawn on celluloid (cel) sheets that where superimposed in order to compose the final frame. *Snow White and the Seven Dwarfs* became an instant success, beating all other films in the box office and becoming the highest grossing motion picture in history, at the time.

Other studios like *Warner Bros* and the *Metro-Goldwyn-Mayer* (MGM) began, in 1935, to make use of watercolor to paint the backgrounds of their animated short films. However, *Disney* had brought it to the next level, using this medium in feature films like *Pinocchio* (1940), *Dumbo* (1941) and *Bambi* (1942).

Watercolor was not the only medium to be used in animation. Gouache, acrylics and oil paint were gaining ground due to their opaque properties and straightforward implementation. Even at *Disney*, after the labor strike in 1941 and the consequences of World War 2, watercolor stopped being the de facto medium of choice. No other feature film by *Disney* was made using watercolors - at least for the next 60 years [Wakabayashi, 2002].

If we observe the role of watercolor in animation up to the middle of the 20th century, an interesting question comes to mind. Why have studios not used watercolors for their character animations? Watercolor was, and still is, widely used in pre-production and look development, also on characters. Nevertheless, its use is mainly avoided in the final films.

> *Disney and other studios alike [back then] did not dare to color its animated characters with watercolor. To do so would require an extensive amount of additional artists to be trained in order to achieve the desired watercolor look. It would have been way too expensive.*[1] (Hans Bacher)

In western animation, there is one particular short film produced by John and Faith Hubley, *Windy Day* (1968) where some shots where produced and animated

[1]Hans Bacher is a highly regarded production designer who worked at *Disney* from 1987-1991 and from 1994-2003 and has over 40 years of experience in the animation industry. This quote comes from a private conversation at the *Nanyang Technological University* in 2014

in watercolor. Even though watercolor was used, a conventional coloring implementation on his designs would require a complex approach to make it look like a living watercolor painting. Because of this, the final rendition of coloring was quite stylized and limited to a few shots only.

Design is key when animating traditional watercolor paintings. Western watercolor paintings tend to be very complex and require a laborious process of layering to attain the desired amount of detail. However, watercolor painting designs in other parts of the world did not embed multi layered processes and details of western watercolor paintings. This was the case for Asian Ink-Wash painting, which focused on expressing both, the spirit of the artist and the subject depicted, in the painting [Lent, 2001].

2.2.2 Te Wei & Ink-Wash Animation

After Chinese animation was thought to be just like Soviet animation at *Annecy* in 1955, Te Wei and the *Shanghai Animation Film Studio* started their pursuit of a distinctive Chinese animation style. Through this new experimentation, the short film *Where is Mama* (Chinese: 小蝌蚪找妈妈) was created in 1960.

The idea of creating ink-wash animation emerged during a brainstorming session on new styles, which would fit the proposed change of direction at the *Shanghai Animation Film Studio*. It was initially received with skepticism since watercolor animation had not been done in foreign countries. A member of the art committee at the *Shanghai Animation Film Studio* back then, quoted in an interview [Shanghai Animation Film Studio, 2003] the reaction of one of the members, who was present at the aforementioned brainstorming session—when considering using watercolors for animation.

> *This is nonsense isn't it? This idea is bullshit. If it can work it should already have been done.* (Qian Yunda)

Nonetheless, Te Wei, being the director, had the final say on the decision. *Where is Mama* was to be done, even with the high risk of failure. So, the studio took on the challenge of China's vice premier, Chen Yi, who stated that one day water and ink painting, mastered by Qi Baishi, would be animated [Zhu and Rosen, 2010]. The production was highly labor intensive due to the nature of watercolors and the development of a new style. But, all the hard work and patience of the animators

paid off. The film, released in 1960, was a significant breakthrough, even though it was only an exhibition work which paved the way for more visually complex films and a unique animation style. After the success of *Where is Mama*, Te Wei, Quian Jia Jun and the *Shanghai Animation Film Studio* continued the study of ink-wash animation and produced the acclaimed film, *The Cowboy's Flute* (Chinese: 牧笛) in 1963. It is a film without dialog, which managed to tell a story almost entirely through animated watercolors.

Figure 2.6: The Cowboy's Flute (1963), © Shanghai Animation Film Studio.

These films were part of the *Golden Age of Chinese Animation* (1955-1965), where studios received financial support and hands-off policy from the Chinese government [Lent and Ying, 2013]. This all stopped once the *Cultural Revolution* (1965-1976) began. However, it resumed later to form a second golden period where Chinese animation blossomed again [Lent, 2001].

New ink-wash animations were produced by the *Shanghai Animation Film Studio* like *The Deer's Bell* (Chinese: 鹿铃) in 1982 but it was not until 1988 with *Feeling From Mountain and Water* (Chinese: 山水情) where ink-wash animation reached its peak. It was Te Wei's final animation before his retirement and a masterpiece on its own [Wang, 2008]. A story told once again without dialog appealing to all audiences, globally.

From 1990 onwards, with the growth of digitalization and globalization, the labor intensive ink-wash animation technique was displaced. The last major animation created traditionally within the style was *The Flight of Wild Geese* (Chinese: 雁阵) in 1990 by Zhong Quan and Wang Gang [Lent, 2001].

Even though no new traditional ink-wash animations are created nowadays without additional help from computers, the significance and historical heritage of ink-wash animation is considered a *Chinese National Treasure* [Shanghai Animation Film Studio, 2003]. It is, by far, the most complex use of traditional watercolors to be achieved in motion.

2.2.3 Contemporary Watercolor Animation

Although traditional watercolor was rarely used for animated films through the rest of the 20th century, watercolors were still employed in a few short independent productions as a medium to paint backgrounds. Some of the best examples are the films by Michaël Dudok de Wit like *The Monk and the Fish* (1994) and *Father and Daughter* (2000).

A remarkable example of animated watercolors came in the form of an experimental sequence in *Fantasia 2000* (1999) by *Disney*: *The Carnival of the Animals* (Figure 2.7). Eric Goldberg and his wife, Susan Goldberg drew and painted 6000 frames to create *Disney*'s first and last cartoon done entirely in watercolors [Ness, 2016]. The *Computer Animation Production System* (CAPS) was used to blend the animated characters with the background. The watercolor production turned out to be labour intensive and expensive, even with the help of CAPS. Unfortunately, the use of watercolors can be easily overlooked for the untrained eye, as the simple coloring style does not present prominent characteristic effects of watercolor.

Figure 2.7: Fantasia 2000 (1999), © Disney.

The last use of traditional watercolor backgrounds from *Disney* in animated feature films came in 2002 with *Lilo & Stitch*. According to Hans Bacher, who was part of the early development as the production designer of the film, they wanted the film to have 'the characteristics of a Hawaiian drink: colorful, friendly and full of details'. Therefore, the production was destined to be made with watercolors, in the style of the old *Disney Classics* (Figure 2.8).

> *By far the greatest challenge was the proposed return to rendering the backgrounds in watercolor. After all, Disney artists had not created an entire feature film using watercolors since Dumbo in 1941* (Ric Sluiter, [Wakabayashi, 2002])

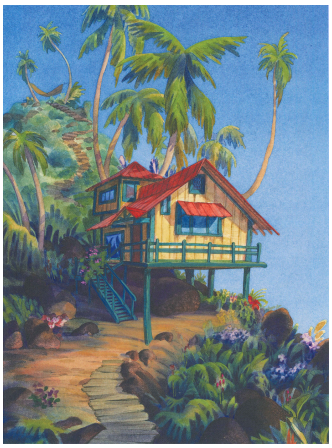
Many of the original backgrounds from *Pinocchio*, *Dumbo* and even the *Silly Symphonies* were brought to the studio as references, where the crew went through an intense year of training in the natural medium. They even reached out to Maurice Noble (a *Disney* veteran who was over 90 years old at the time) to consult on his workflow and ask for advice.

Lilo & Stitch was a complete success, creating a franchise with three direct-to-video sequels and a television series. Nevertheless, traditional watercolor was not used in them and was replaced by less complex or digital media. In fact, even though watercolor backgrounds had been revived at *Disney* with many new artists trained in the medium, no other film featuring watercolor was done.

With the adoption of digital workflows in the animation industry, the possibility of creating watercolor-looking paintings using computers became real. A few productions promptly took advantage of these new technologies. One remarkable seminal example being the animated feature film *My Neighbors the Yamadas* (1999) by *Studio Ghibli*. The film has a distinctive simple comic design, which helped the implementation of watercolors. Even the characters were made to resemble a watercolor look, although they did not have prominent watercolor effects (Figure 2.9). *My Neighbors the Yamadas* marked the beginning of the next generation of watercolor films, which rely on computers to aid in the achievement of their look—more about computer generated watercolor can be read in the next section.

Once the animation industry adopted digital workflows, the use of traditional watercolors in films became obsolete. However, this does not mean that watercolor was stopped from being used in independent productions. A few short independent productions like *Naked Branches* (2007) by Will Kim continued to make use of the traditional medium in animation. Other films have also partially featured the traditional medium like *Madagascar Journey Diary* (2010) by Bastien Dubois.

Gouache



Watercolor

Figure 2.8: Lilo & Stitch (2002), © Disney.

Figure 2.9: My Neighbors the Yamadas (1999), © Studio Ghibli.

A remarkable and surprising feat was accomplished by Anders Ramsell, with his artistic watercolor take on *Blade Runner - The Aquarelle Edition* (2013). For this animated production, he painted 12,597 watercolor frames over a two-year period, to produce his 34-minute tribute to the film *Blade Runner* (1982). While each painted frame was tiny (Figure 2.10), the production still required a gargantuan effort. Moreover, by painting at such a small size, the production was given a dreamy look, which let the pigment interaction with the water and paper fill in the details. His next film *Genderness* (2016) also applied the same watercolor technique, though at a slightly larger painting scale and shorter duration span (7 minutes).

Figure 2.10: The size of each painting (1.5x3cm), © Anders Ramsell.

It is of importance to highlight that the digital revolution has also created new possibilities for traditional watercolor in animation. Short films like *Galileo* (2008) by Kori Michele Handwerker and the music video for *Breakbot –Baby I'm yours* (2014) by Irina Dakeva made use of digital rotoscoping techniques and traditional watercolor paint, to create wonderfully crafted animated short productions.

Figure 2.11: Breakbot (Baby I'm Yours feat. Irfane) (2014), © Wizz.

2.2.4 Future

The use of traditional watercolor in animation is unlikely to ever become mainstream. With so many simpler traditional madia and the digital implementation of paint taking over the role of traditional paint in production, watercolor is destined to become a medium for the nostalgic and purists. Nevertheless, watercolor animation will not disappear.

There are still short productions in recent years, like *No Fish Where to Go* (2014) by the *National Film Board of Canada*, *An Object at Rest* (2015) by Seth Boyden, *Out of My Mind* (2016) by William Anugerah and *Drops* (2017) by Sarah Jungen and Karsten Hoop, which are keeping the traditional medium afloat—with the help of digital compositing tools. Traditional watercolor backgrounds are also still present in feature films like *Ernest & Celestine* (2012) by *La Parti Productions* and *Les Armateurs* and *The Tale of Princess Kaguya* (2013) by *Studio Ghibli*.

With the pressure of the industry nowadays to achieve high quality productions on a tight budget and production time-frame, we might not see a traditionally animated watercolor feature film anytime soon. However, with the critical success of *Loving Vincent* (2017) by *BreakThru Productions* and *Trademark Films*—the first animated oil paint feature film—who knows what the future might bring. One thing is certain, independent short productions using traditional watercolors will stay, keeping the medium alive in animation and honoring this century-old craft.

While the use of traditional watercolor in feature films is potentially fated to disappear, the watercolor look in films is still present. Many technological advances in computer generated imagery (CGI) have enabled a convincing representation of the watercolor medium, which permits a much faster and straightforward implementation of the look. These latest technological advances are making feature length, watercolor-looking films a viable option within the entertainment industry. Furthermore, a series of new productions have been created with an entirely digital watercolor look.

2.3 Computer Generated Watercolors

The coming of age of computers opened the opportunity to synthesize watercolors, digitally. However, digital workflows have been a controversial topic for some professionals (working traditionally) and consumers alike. Therefore, since this section focuses on computer generated imagery (CGI), it is important to emphasize the following statement.

> *The term CGI is a misnomer—the computer doesn't generate the images. That would be like calling traditional animation pencil-generated imagery. No matter what the tool is, it requires an artist to create art.*
> (John Lasseter, [Smith, 2002])

The synthesis of watercolor has been researched since the early 1990s and its techniques and algorithms have found vast implementations, especially within painting and photo manipulation software. This chapter will cover what has been achieved artistically through computer generated watercolor, on still paintings and in motion/animation.

Figure 2.12: It's artificial?, © Martin Hanschild.

2.3.1 Digital Watercolor Images

Nowadays, the watercolor brush is an established tool in many painting software. With widespread availability, digital watercolor artwork is being painted everyday. However, few professional watercolor artists are making the shift to digital workflows, as there is little demand for digital watercolor artwork from serious clients. Instead, digital watercolor painting is adopted by amateur and hobbyist painters, who take the most advantage of the digital conveniences offered. Additionally, it has also found use in artwork that is not meant to be shown in printed forms.

Watercolor images can also be synthesized from photographs through watercolorization filters that are featured in photo manipulation applications. These filters are automated image processing algorithms that run on the original pixels, transforming a photorealistic image into a watercolorized image. This method does not even require painting skills and has found wide adoption with casual users that want to add an artistic touch to their photographs.

Generating watercolor images is at the core of the research presented in this thesis, therefore, existing applications available in both watercolor painting and filtering will be surveyed next.

Watercolor painting

The physical behavior of watercolor on paper has been studied for decades. It was David Small who wrote the first technical paper [Small, 1991], which marked the beginning in digital watercolor painting. The technology and algorithms have come a long way since then, resulting in many applications offering robust simulations of watercolors e.g., *Artrage (Ambient Design)*, *Painter (Corel)*, *Expresii* and *Rebelle (Escape Motions)*. Watercolor painting is also present in mobile platforms e.g., *Procreate (Savage)*, *Sketch (Adobe)*, *Paper (FiftyThree)*, *Auryn Ink (Auryn)* and *Artrage (Ambient Design)* for mobile.

Although widely available in painting applications, watercolor is not as popular as other digital painting tools, making it rather uncommon in the bulk of digital paintings that are being created every day. The versatility of a watercolor brush to replicate traditional watercolor varies from application to application. However, some applications, especially in desktop platforms, offer a vast amount of control over each single brush stroke. This can be useful to produce a faithful synthesis of the medium (Figure 2.12), giving a prominent selling point for a painting application. However, the vastness of parameters, knobs and settings might also drive people away.

(a) My Bunny!, © Nino Aptsiauri (Rebelle)

(b) 離珠, © Shuen Leung (Expresii)

(c) Trees and Hills, © Stephen Berry (ArtRage)

Figure 2.13: Digital watercolor paintings by traditional watercolor artists.

It is rare to find complex examples of digital watercolor paintings that use different software, but remarkable examples have been done by artists with a traditional watercolor background (Figure 2.13). The first images, Figure 2.12 and Figure 2.13a, were painted in *Escape Motion*'s *Rebelle*, probably the most advanced commercial application for watercolor painting. It features a proprietary paint engine, which is responsible for the characteristic watercolor effects. Figure 2.13b was painted in *Expresii*, which specializes in fluid media and excels in eastern watercolor (ink-wash) synthesis. The software is based on the research of Chu and Tai [2005], which relies on computational fluid dynamics for the stroke synthesis. The last example (Figure 2.13c) was painted using *Ambient Design*'s *ArtRage*, which is

a more general painting software than the previous examples. It features a procedural approach to synthesize the watercolor effects and makes use of smudging and blending to enhance the watercolor look. Finally, it is important to remark that all the previously featured painting software rely heavily on the substrate (paper) profile for the synthesis of the watercolor look.

While the previous examples (Figure 2.13) represented an excellent synthesis of the traditional medium, digital watercolor painting is still a complex and challenging endeavor. The watercolor brushes and the substrate offer extensive customization to provide the flexibility of the traditional medium. Therefore, to produce a faithful synthesis of watercolor, a skilled artist in the medium with a good attention to detail is required—who is also comfortable using digital devices.

Painting digitally offers the advantage of independent layers, undoing and non-destructive manipulation, but it can also involve a lot of patience and extra effort to make the painting look, as if it would have been painted traditionally. The finesse and accidental details of water and paint are much harder to synthesize, even using computational fluid dynamics. Because of this, digital watercolor paintings are not necessarily faster to produce than its real counterpart and certainly less intuitive for experienced watercolorists.

Digital painting provides an excellent low-level of control over the watercolor synthesis, but this entails the need for skilled artists to produce a faithful result. For the less skilled individuals, watercolor synthesis can be achieved by using image processing in the form of filters.

Watercolor filtering

One of the first synthesized images which showed a resemblance to traditional watercolor (Figure 2.14) was produced in the seminal paper by Curtis et al. [1997]. Using post-processing algorithms, this approach paved the way for the various techniques and algorithms which followed (Chapter 3).

Nowadays, the majority of photo manipulation applications feature some sort of watercolor synthesis from pixel images—in the form of filters in desktop and mobile platforms e.g., *Photoshop (Adobe)*, *PhotoDonut (Xycod)*, *Waterlogue (Tinrocket)* and *BeCasso (Digital Masterpieces)*. However, these filters are rarely used as a standalone solution for emulating a watercolor look, since the filtered results are heavily constrained by:

- *Filter design*: algorithms manipulating the image data.
- *Source image*: photograph or computer generated image.

Filter design. In most cases, the algorithms are hard-coded (fix), but can be manipulated via a set of exposed parameters that are tweaked by the user to affect the end result. This might give a high-level of control over certain qualities and effects of watercolor, but results still remain predictable and recognizable—reducing the originality and creative values. To illustrate this argument, we can run some photographs through watercolor filters and analyze the outcomes (Figure 2.15).

For this visual experiment, all photographs were first reduced to a smaller resolution of 582x863 pixels. Then, each example was processed by the default watercolor filter found in *Adobe Photoshop CC 2017*, *Waterlogue 1.3.1* and *BeCasso*, respectively. *Photoshop* was chosen for being the de-facto choice for photo manipulations by professionals, *Waterlogue* was chosen for being a popular proprietary watercolor filtering application and *BeCasso* was chosen for being a mobile filtering application developed from watercolor research in academia [Bousseau et al., 2006; Semmo et al., 2016; Wang et al., 2014]. All applications, though featuring a watercolor filter, produced completely different results.

Photoshop's conservative take on watercolor hardly resembles the traditional paint and the spontaneity of watercolors. Additionally, the filtering approach considerably darkened the resulting watercolorized imagery and randomly ignored important edges (e.g., the cable in the third example). *Waterlogue* is able to do a good job at mimicking some imperfections of the human hand. It also automatically adapted the illumination choices, the watercolor palette and created random color bleeding effects and abstractions. Whether these changes are desired is a constraint of this approach, as there is no way to art-direct the filtered result. *BeCasso* preserved the original colors of the source photography, but also retained significant detail that might not be considered by an artist. There is also no color bleeding, observable inked lines in questionable places (clouds of example 2, Figure 2.15)

Figure 2.14: Automatic image "watercolorization". © Curtis et al.

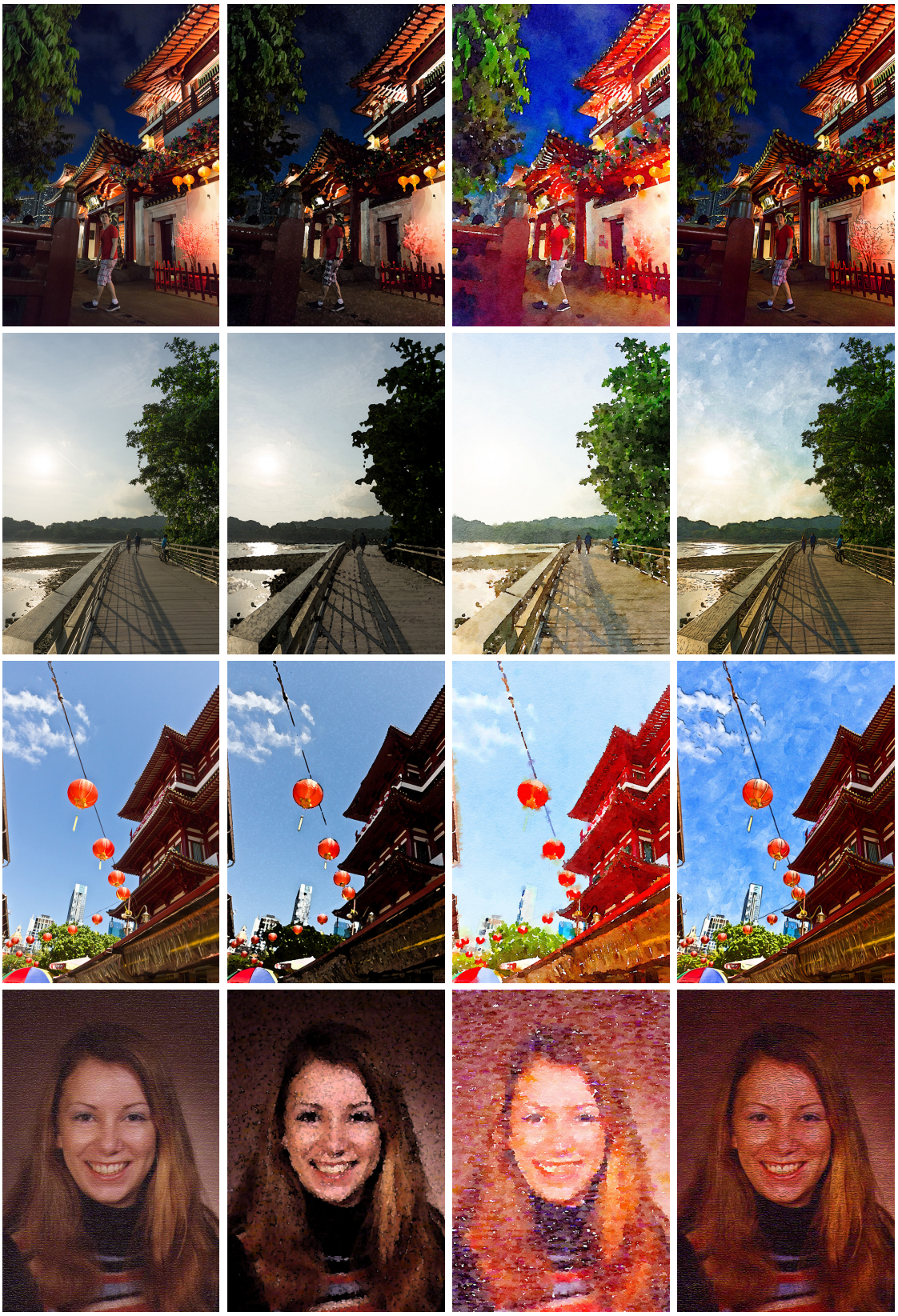


Figure 2.15: Photography filtered through different commercial applications.

and background watercolor patterns that repeat themselves in each filtered case (observable in the sky of example 2 and 3, Figure 2.15).

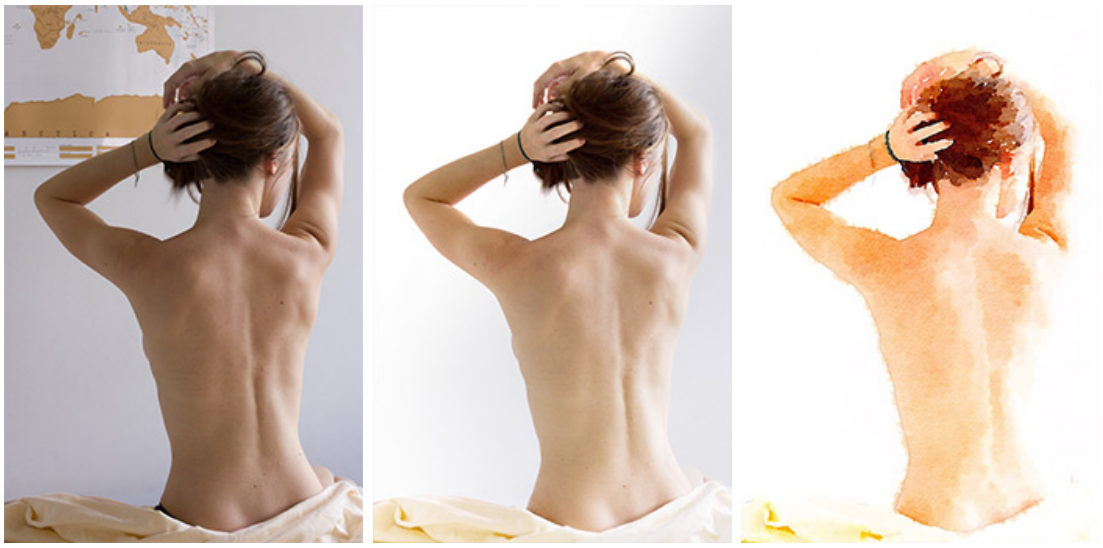
In general, *Waterlogue* and *BeCasso* do a reasonable job at synthesizing certain aspects of watercolors. Therefore, both applications remain quite popular with watercolor enthusiasts and photo filtering communities. Nevertheless, *Waterlogue* suffered from the absence of control on the filter's parameters, which gives *BeCasso* an advantage. While the initial result in *BeCasso* might not be as watercolor-looking as *Waterlogue*, it offers multiple levels of control in image-space to thoroughly art-direct the stylized result.

Each filtering approach presents a defined look with patterns that are easily identifiable, as image filtering is constrained by its filter design. In cases where the filter design is fixed, the only way to amortize the patterns is by offering levels of control over the stylization. However, the success of filtering approaches highly depend on the source images, which might contain visual clutter that the algorithms are not accounting for—and which an artist would never depict in a painting.

Source image. The source images in filtering approaches have an influential role in the filtered result. This is easily observed in the portrait photograph seen in the fourth example (Figure 2.15), which has a textured structure from the scanned paper that heavily affected the filtered result. These patterns would have never been painted by an artist. Therefore, for a successful synthesized watercolor painting, the source image needs to possess only the elements that an artist would paint. Additionally, source images tend to have many visual elements which a painter would ignore—in favor to the essence of the subject.

The visual clutter of a source image can be taken into account at the moment of creation or it can be manipulated to the artist's intent. If the source image is arranged or manipulated to aid in the creation of a synthesized watercolor painting (Figure 2.16), a better synthesized painting can be created. The redundant visual elements are removed, keeping the subject at the essence of the artist's interpretation. These prior steps, together with the right filter design, enable the final image to bear a much stronger resemblance to a watercolor painting and the desired vision of the artist. Prior image manipulation can significantly aid the filtering process, but this work can take extensive time, skills and effort, which reduces the advantages that a filtered synthesis would provide, over manually painting the subject in the first place.

> *No artist will just take a photograph and stylize it, it is bad design.*
> (Joe Cartwright [2015])



Original photograph Manipulated photograph Watercolorized photograph

Figure 2.16: Manipulation of a source image to produce a better synthesized result.

In animation, where the subject is intrinsically designed, this does not present an obstacle. There has always been a desire to achieve a storybook feel in animated productions, as the visuals in most of storybooks have been painted with watercolors. Therefore, some productions have ventured into synthesizing watercolors in animation, which are surveyed next.

2.3.2 Digital Watercolor Animation

Creating traditional watercolor animation requires a lot of trained artists, in the specific watercolor style, who need to keep every frame consistent and control how the medium behaves over time. Because of this, few productions have ventured to use the medium in motion, as discussed previously (Section 2.2). Once computers enabled digital tools for animated production, technology was also used to create watercolor-looking films and games. However, the methodology to synthesize watercolor in animated films and games are different.

In animated films, each watercolor frame (image) can take indefinite time to process, enabling any desirable computation to be processed. This methodology is called *pre-rendering* or *offline rendering*. In games, each frame needs to be calculated interactively while the previous frame is displayed—requiring highly optimized algorithms that run as fast as possible. This methodology is called *real-time rendering* or *online rendering*.

The results from pre-rendering and real-time rendering vary significantly due to the different times that both methodologies can afford to process watercolor imagery. Therefore, this section is divided into pre-rendering and real-time rendering of digital watercolor in motion, featuring animated films, short-films and games.

Figure 2.17: Experimental animation painted in *Expresii*, © Angela Wong.

Pre-rendering

Pre-rendering, also known as offline rendering, is found in form of films, short films and experimental sequences. One of the first to use a computer to synthesize watercolor in motion was the short film *Fishing* (1999). *Fishing* was an experimental short film created at *Pacific Data Imagery* (*PDI*) in which Cassidy Curtis synthesized a watercolor look from 3D objects in the computer.

Curtis achieved a successful (although limited) watercolor look (Figure 2.18) by rendering a shadow matte of a 3D scene and post-processing the result using filters and noise synthesis [Curtis, 2015]. This technique achieved convincing results that could most likely be implemented nowadays in real-time, as well. However, the technique can only be applied to its specific flat style of watercolors.

On the same year, *Studio Ghibli* released their film *My Neighbors the Yamadas* (1999), directed by Isao Takahata, which also synthesized a watercolor look in a comic strip graphic style (Figures 2.9 and 2.19). Takahata's ambition concentrated on seamlessly integrating the characters with their environment, which presents a real challenge in hand drawn animation. The style of the film was inspired by the short film *Crac* (1981) by Frédéric Back.

Figure 2.18: Fishing (1999), © Pacific Data Imagery.

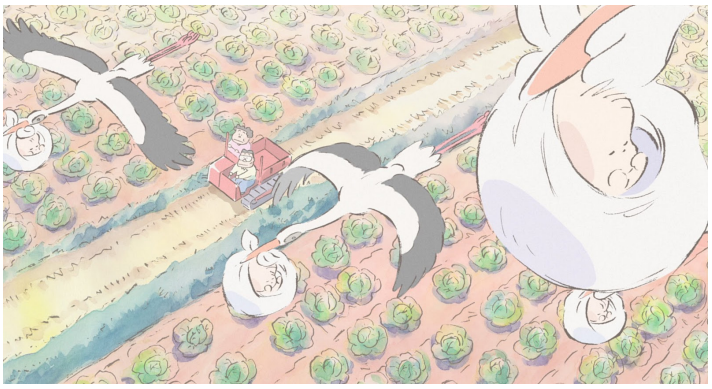


Figure 2.19: My Neighbors the Yamadas (1999), © Studio Ghibli.

My Neighbors the Yamadas included traditionally painted backgrounds with computer aided characters. First, the characters were hand-drawn on paper to create the outline. A second hand-drawn pass was then created, tracing the outline of the first pass and closing regions of color. After scanning both passes into the computer, the regions of the second pass were filled with color using a software specifically developed for the film. The outline, created on the first pass, was then superimposed over the color to obtain the final looking image, which was composited on top of the traditional watercolor backgrounds. There are a few sequences which also make use of 3D graphics. To stay truthful to the look, a traditionally painted watercolor texture was projected onto 3D geometry. This enabled slight camera movements through the environment without significant texture distortion. Additionally, a geometry outline was rendered to simulate the hand-drawn lines. These outlines were then jittered, to simulate slight hand tremors over time.

Overall, the characters in the film do not contain many distinctive effects found in watercolor. However, the hand-drawn outline and the slight irregularities in the colored regions, together with the homogeneous colors between the traditional backgrounds and the characters, make the final rendition quite successful at synthesizing a watercolor style in motion.

In 2001, the Korean live-action film *Wanee & Junah* (2001) surprised the animation community by featuring seven minutes of watercolor styled animation (Figure 2.20). Little is known about the techniques used to produce the watercolor look of the characters. However, it is known that the animation was rotoscoped (traced) from live-action footage and that the backgrounds, which are static, are painted traditionally. The watercolor implementation on characters is convincing at distance but shows significant flaws when they are close to the camera, which, combined with the rotoscoped animation, touches the uncanny valley[2].

[2]Used in reference to the phenomenon whereby a computer-generated figure or humanoid robot bearing a near-identical resemblance to a human being arouses a sense of unease or revulsion in the person viewing it. (Oxford Dictionaries 2015)

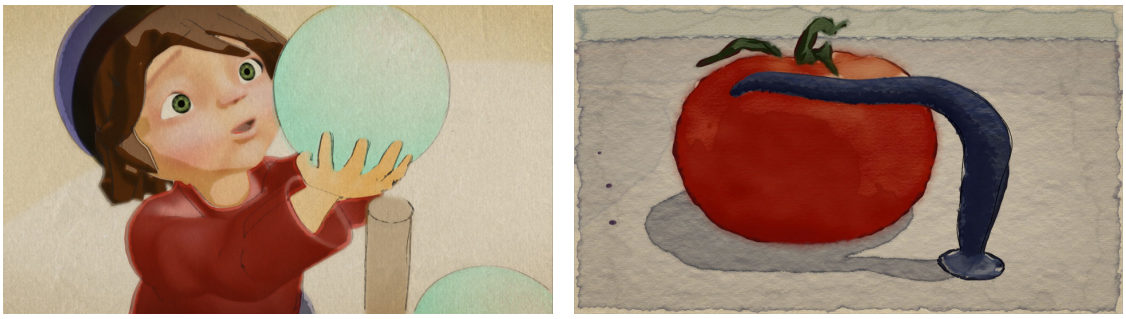
Figure 2.20: Wanee & Junah (2001), © Kim Kwang-su.

Another short-film which featured a watercolor look is *The Girl Who Cried Flowers* (1999) by Umesh Shukla. The film features traditional watercolor backgrounds together with 3D animated characters (Figure 2.21). While the exact technique used to achieve the look is unknown, texture mapping of traditional watercolor images can clearly be observed. The characters are rendered through an in-house watercolor engine named *Beatrix* [Shukla, 2015]. Unfortunately, they present distracting temporal coherence problems and it was overall too detailed, distancing itself from the hand-painted watercolor environment.

Other experimental short-film productions were produced once the technology developed by Thomas Luft (Section 3.3) became accessible to 3D animation filmmakers through *aQtree*. The software was used in close collaborations with the *Institute of Animation* at the *Filmakademie Baden-Württemberg* to produce *Kieselstein* (2007), by Ellen Hoffmann (Figure 2.22a), and *Love.Leech.Tomato* (2007) by David Maas (Figure 2.22b). Unfortunately, probably due to the complexity of the watercolor synthesis used by *aQtree* and its closed-source nature, no further films continued using this technology.

A watercolor synthesis was partially used by Mauro Carraro in his mixed media NPR productions *Matatoro* (2010), *Hasta Santiago* (2013) and *Aubade* (2014). While the pre-production model turntables and look development present charac-

Figure 2.21: The Girl Who Cried Flowers (1999), © Umesh Shukla.



(a) Kieselstein (2007), © Ellen Hoffmann **(b)** Love.Leech.Tomato (2007), © David Maas

Figure 2.22: Animations done with *aQtree*.

teristics of watercolors, these take a secondary role in the final aesthetic of the films. The look in Cararro's films was achieved through clever use of offline compositing techniques, but not much is shared regarding the exact procedures.

Recent feature films presenting a watercolor look have been produced in France, with *Ernest & Celestine* (2012, Figure 2.23) and *Big Bad Fox & Other Tales* (2017, Figure 2.24). These films were based on children's books, *Ernest & Celestine* by Gabrielle Vincent and *Big Bad Fox* by Benjamin Renner. Because of this, they aimed to stay truthful to the watercolor and ink style featured on Vincent's and Renner's illustrations. The productions could have been made completely with traditional watercolors, but the budget did not allow for such experimentation. Therefore, in the case of *Ernest & Celestine*, the production relied on mixing traditional watercolor and ink with digitally painted animation. For the *Big Bad Fox & Other Tales*, a completely digital workflow was implemented, which took advantage of scanned traditional watercolor washes with effects.

To use as much of the traditional medium as possible, the still parts of the image and the animated ones were separated from each other. The parts without movement, which were mostly environments, were painted traditionally in *Ernest & Celestine*. Almost a thousand environments were hand-drawn with pencil to later be traced with black ink. These inked environments were then scanned into the computer and printed again in order to be used as a tracing template for the watercolor painting. Watercolors were applied on a new sheet of paper and once finished, the colored environments were scanned again, composited with the ink lines and color corrected in *Photoshop* to match the original illustrations. In the case of *Big Bad Fox & Other Tales*, the backgrounds were produced entirely in *Photoshop*, relying on masking traditionally painted watercolor effects on top of digitally painted elements to achieve the right look.

Everything animated was instead created using a computer in both films, but a lot of hand labor was still required. Each frame was hand-drawn in *Adobe*'s *Flash*, keeping gaps between the strokes to resemble a sketched look. Therefore, the paint-

Figure 2.23: Ernest & Celestine (2012), © La Parti Productions and Les Armateurs.

ing was also filled by hand, as there were no closed regions to be automatically filled with color. To simulate the final watercolor look over the digitally painted animation, compositing techniques where used in both productions to emulate pigment turbulences and add some graininess on top of the solid colors. Backgrounds and animations then underwent a final color correction to seamlessly integrate the backgrounds with the computer animation.

Two more films worth mentioning here are *The Tale of Princess Kaguya* (2013) and *Cheatin'* (2013). *The Tale of Princess Kaguya*, directed by Isao Takahata, the same director of *My Neighbors the Yamadas* (1999), concentrates on telling the story through a sketched style [Miki and Satou, 2015]. Of particular interest is how the animation is pushed to new boundaries, modifying the sketched frames according to the mood of what is happening in the story. This includes extensive use of traditionally painted watercolors and digitally painted characters. While the characters did not have the characteristic effects of watercolors, they still were carefully painted to fit into the watercolor backgrounds. *Cheatin'*, directed by Bill Plimpton, was also drawn by hand with pencils, following a sketched style. However, the film was completely colored digitally using *Adobe Photoshop*. The watercolor look was achieved by overlaying hand-painted traditional watercolor textures over the colored backgrounds.

Figure 2.24: Big Bad Fox & Other Tales (2017), © Folivari and Panique! Production.

(a) Dog and Butterfly (2014), © Wayne Unten **(b)** Madeleine (2015), © Nic Debray

Figure 2.25: Experimental short animations with a watercolor look.

Several animated clips and experiments have also been done, simulating the traditional watercolor medium. These examples can be found online and feature different degrees of success, from basic watercolor shader experiments to polished animation tests. An in-depth analysis of each one of these examples is out of the scope of this thesis, as there is little information about them—most of them being at experimental stages. However, there are three remarkable examples of watercolors in motion that present a well defined watercolor aesthetic in experimental short clips e.g., *Dog and Butterfly* (2014) by Wayne Unten (Figure 2.25a), *Madeleine* (2015) by Nic Debray (Figure 2.25b) and a squirrel animation test done by Angela Wong (Figure 2.17). *Dog and Butterfly* was entirely painted frame by frame in a software called *TV Paint* (10.5 Pro) with digital brushes. *Madeleine* was painted traditionally, but watercolor textures were animated digitally to follow the pencil animation. The squirrel test was also painted frame by frame in *Expresii*, an eastern watercolor painting software (Section 2.3.1).

Finally, Chinese ink-wash has also been synthesized digitally in animation with relative success e.g., *The Way* (2003), *Ode to Summer* (2003), *A Hero Who Commands the Sky* (2005), *Movement and Stillness* (2006), *Way of the Mantis* and *Passion of Ink and Wash* (2011). However, these short films are not going to be reviewed as the research presented in this thesis concentrates on synthesizing western application of watercolors.

2.3.3 Watercolors in real-time

Real-time rendering, also known as online rendering, is mostly found in the form of 3D games and experimental render tests. Compared to pre-rendering, few examples exist that try to synthesize watercolors. Because in real-time applications, each frame needs to be computed interactively, while the previous frame is be-

ing displayed. This means that every image needs to be created in a matter of milliseconds, which makes an actual synthesis even more challenging.

There are 2D games which feature a watercolor look in motion e.g., *Child of Light* (2014) by *Ubisoft Montreal* and *Cuphead* (2017) by *Studio MDHR*. However, 2D games have the advantage to pre-render any watercolor-looking images and display them interactively in the form of sprites. Therefore, 2D or 2.5D games are excluded from consideration.

Of the few examples of real-time rendered watercolor synthesis in motion, the game *Okami* (2006) by *Clover Studio* plays a seminal role (Figure 2.26). While there have been non-photorealistic 3D games before, *Okami*, which has also been remastered and can be found as *Okami HD*, was the first game to accomplish a watercolor resemblance. It features an ink-wash style with strong inked outlines and extensive use of watercolor textures which are cleverly placed to resemble the traditional medium.

While *Okami* is a great example of ink-wash rendering in real-time, many characteristic effects present in western watercolor renditions are not addressed. Additionally, most of the addressed effects are manually painted in textures which are mapped to the geometry.

More recently, the indie game *Beyond Eyes* (2015) by *Tiger & Squid* featured a painterly look resembling watercolors (Figure 2.27). While no characteristic effects of the watercolor medium are simulated, the vibrant use of colors, the hand painted watercolor-looking textures and a focused reveal of the environment does resemble a watercolor illustration.

There are also experiments online, which attempt synthesizing watercolors in real-time. One remarkable example is *Project MIS* (2013) by *AlfierTheWise* (Figure 2.28). Unfortunately, no documentation is available as to how the look was created. However, upon close observation, the use of image-filtering can be intuited.

Figure 2.26: Okami HD (2012) by Clover Studio.

Figure 2.27: Beyond Eyes (2015) by Tiger & Squid.

Especially the use of textures with pigment turbulence, which are overlaid over the rendered frame to recreate pigment density variation; and an edge detection filter, which also enhances the edges to create outlines and emulate the darkened edge effect found on traditional watercolors.

Apart from *Beyond Eyes* and *Project MIS*, it is rare to find other real-time implementations of stylized rendering that resemble a watercolor look. Therefore, *Beyond Eyes* received considerable praise at the the *Game Developers Conference* (GDC) and at the *Electronic Entertainment Expo* (E3) in 2015, because of its distinctive look. However, it did not actually synthesize any characteristic effects of the traditional medium, itself. Up to the point where I started my research in real-time watercolor rendering of 3D objects and animation with enhanced control (August 2014), almost no animated productions featured a real-time synthesis of watercolor.

This niche presented was an immense opportunity to offer a renewed visual look of watercolors to the entertainment industry, which made use of real-time technologies to present stylized results immediately and enable interactive art-direction. In the next chapter, a survey of previous research done towards a watercolor synthesis is presented, by reviewing the related work done in stroke-, image-, object-space.

Figure 2.28: Project Mis (2013) by AlfierTheWise.

3. Related Work

The synthesis of watercolor imagery has been extensively studied over the past decades, starting from a seminal paper that studied the spread of pigment and water on a simulated paper canvas and calculated its results in a supercomputer with 16k processors [Small, 1991]. This physically-based approach, though limited in synthesizing all the various effects and characteristics of watercolor, was a breakthrough which paved the way for future research in the synthesis of traditional painted media.

This chapter will survey the related work available in the literature. Over the past two decades, various approaches to digitally synthesize watercolor have been proposed, which facilitate the production of watercolor paintings and animations. These approaches can be organized into three main categories according to their respective input data: (1) stroke-space; (2) image-space; and (3) object-space.

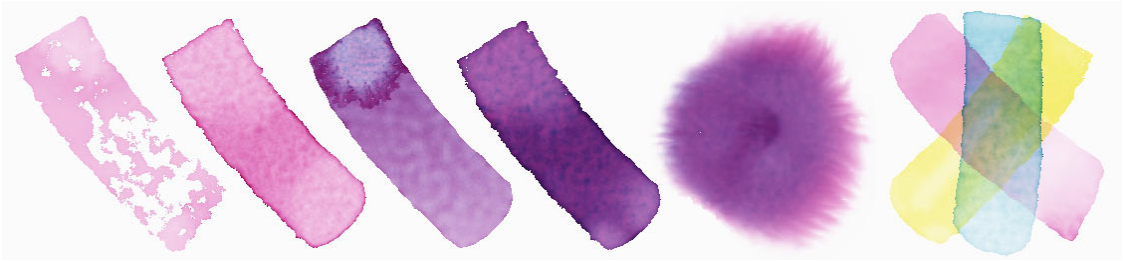
Each category has its own specialized application, however, they often tend to interrelate to enhance the final watercolor simulation, producing hybrid-space approaches. The research presented in this thesis is conceived to render 3D animated geometry as synthesized watercolor animations. While the input data is in object-space, our output data is found in the form of images, which can be used as input data for further image-space processing towards our synthesis. Therefore, the approach presented in this thesis is considered a hybrid object-space approach.

Most approaches found in these categories are concerned with the synthesis of characteristic effects found in watercolors. Therefore, each category/group of approaches will be surveyed and introduced. Finally, a brief overview of other relevant papers is given, which covers additional work capable of emulating watercolors. However, this survey will focus on approaches that could permit a real-time synthesis of watercolors, applied to 3D generated imagery.

3.1 Stroke-space Approaches

Stroke-space approaches are mostly used in painting applications by taking a digitally drawn line as input and synthesizing the thickness of a stroke and behavior of the pigments along a specific area of effect. In ideal cases, stroke-space approaches operate at constrained resolutions, limiting the necessary computation and allowing real-time and interactive placement—mimicking the immediacy of painting with natural media. Because of this, even physical approximations of watercolors can be applied, to a certain extent, in stroke-space approaches for synthesizing watercolor. In this case, each brush stroke goes through its respective physical simulation until a threshold time has passed, presenting the simulated result. Remarkable examples came early on, which analyzed and calculated the spread of pigment in simulated paper [Curtis et al., 1997; Small, 1991].

The first approach specifically addressing the effects of watercolor was proposed by Curtis et al. [1997], who laid out a model of the underlying procedures of water and pigment advection. The approach has an in-depth focus on imitating the physical behaviors, keeping in mind the aesthetic results of watercolor. Through this model they could synthesize various characteristic effects successfully (Figure 3.1).



Dry-brush, edge darkening, backruns, granulation/separation of pigments, flow patters and color glazing.

Figure 3.1: Simulated watercolor effects, Curtis et al. [1997].

The watercolor synthesis is done by creating a set of washes that have undergone a fluid simulation. This fluid simulation is made using a three-layer model. The first layer represents the advection of water and pigment. The second one handles the pigment absorption and desorption done by the paper. The third and last layer is used to enable backruns, which need to be treated as a special case. Once these washes have been calculated, they are composited using the *Kubelka-Munk* color model [Haase and Meyer, 1992] to generate the final color glazing effect. The synthesis was heavily influenced by a generated paper texture, which was simplified for simulation purposes. Instead of having the complexity of intertwined fibers and exact surface properties, it was generated through a procedural texture.

This approach was integrated into a painting application, where the initial pigment position was interactively placed by the user. Then, the simulation took on

to iteratively calculate all the glazes, composite them and create the finished painting. A filter was also proposed to synthesize a watercolor painting from images, but this function was quite limited in terms of automation. The color pigments had to be chosen manually and the images went through a semi-automated two-stepped process featuring color separation and brush stroke planning. However, the results of the filtering approach could still produce convincing imagery (Figure 2.18).

The paper of Curtis et al. bears importance by being the first one to specifically attempt to synthesize watercolor effects and create a filter which simulates them. While the simulation happens in stroke-space and approximates watercolors through physical behaviors, this semi-automated filter is the first to partially implement an image-space synthesis of watercolors.

Using the same paper texture model, a procedural approach (instead of a physical approximation) to synthesize watercolors was proposed by Kalnins et al. [2002], which focused on stroke-space rendering algorithms on 3D geometry (Figure 3.2). While the implementation of watercolor was not their main focus, a specific combination of features enable them to procedurally reproduce simple watercolor-like renders.

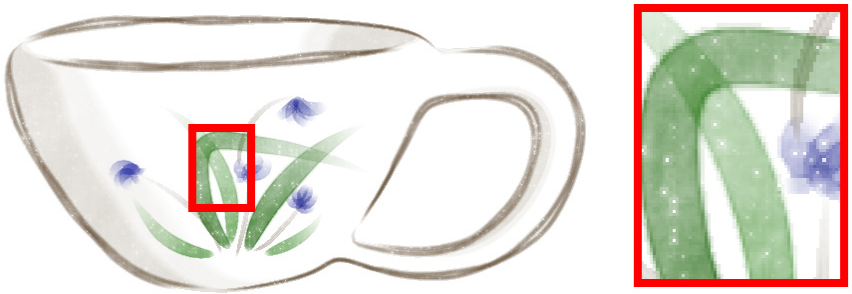


Figure 3.2: Stroke-space synthesis, Kalnins et al. [2002].

The base layer of the render is a ramp-based toon shader which changes gradually from light gray to white. The watercolor-looking brush strokes where painted on the model creating a base path along the geometry which is represented as a triangle strip. The watercolor effects were synthesized using pixel/fragment shaders. The substrate granulation is based on the paper texture's height, emulating basic pigment accumulation on the valleys of the paper by modifying the alpha values. The edges of the triangle strip were also darkened to emulate the darkened edges effect of traditional watercolor. Since strokes might be placed differently at varying distances from the camera and these are represented by geometry, further levels of detail had to be created, to blend the triangle strips relative to the camera position.

The work presented by Curtis et al. and Kalnins et al. paved the way for physical and procedural approaches towards synthesizing watercolor in stroke-space. Significant research has since then superseded these approaches, offering a success-

ful synthesis of fluid media. Physical approximations now allow each brush-stroke to be intricately controlled to achieve most desired effects and a characteristic watercolor look [Chu and Tai, 2005; Van Laerhoven et al., 2004; Van Laerhoven and Van Reeth, 2005; You et al., 2013]. Nonetheless, it can be difficult for an artist to understand and interpret the physical attributes, which these simulations offer. Therefore, procedural and example based approaches [DiVerdi et al., 2013; Lu et al., 2013] have offered simplified alternatives, which can also synthesize characteristic effects that integrate into the overall aesthetic of a watercolor artwork.

Stroke-space watercolor syntheses provide a faithful aesthetic and extensive artistic low-level 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. Additionally, the research presented in this thesis focuses on the interactive stylization of 3D animations, for which an artist cannot manually draw each image. Stroke-space watercolor synthesis could only be directly used if strokes were placed in 3D space [Quilez, 2016; Schmid et al., 2011], deformed as 3D objects and stylized. Unfortunately, this approach brings additional challenges, as it requires custom assets, deformers and an unconventional rendering pipeline. Instead, stroke-space watercolor synthesis can be used indirectly to paint watercolor-looking textures which are then mapped onto 3D objects. However, these textures would remain static and are prone to perspective and geometric distortion, without further processing. Image-space and object-space simulation techniques are better suited for the objective and are presented next.

3.2 Image-space Approaches

Image-space approaches are used mostly in filtering and image processing applications, by taking pixels on a two-dimensional image plane as input and performing operations on them. These operations may happen by only using single pixel operations (e.g., saturation enhancements); by using algorithms and help from external image-space data such as kernels or other images (e.g., edge detection, blending); or by using algorithms driven by a set of rules from other computer vision algorithms, which aid in processing the final image (e.g., color adjustments based on a database of images).

Approaches in image-space could also be considered an extension of stroke-space approaches, but taking an entire image as its input. However, physical simulation to an entire image involves a substantial increase in computation time, compared to processing only a brush stroke. Additionally, the final outcome becomes difficult to control and to art direct, as a naïve application would heavily distort and smear the image (Figure 3.3). A more controlled application could be possible through the use of masks and a prior definition of parameters, as presented by Curtis et al. [1997] (Figure 2.14). Further processing such as segmentation would provide better results, but this would add more complexity to the computationally expensive physically-based approaches. Therefore, procedural approaches are generally better suited for image-space synthesis—especially considering our application in real-time rendering, where each frame (image) needs to be processed in fractions of a second.

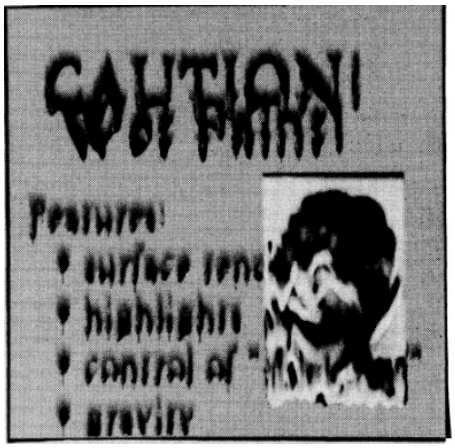


Figure 3.3: Physical simulation, Small [1991].

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.

Procedural image-space synthesis of watercolors has been studied by various researchers, proposing approaches that feature watercolor-looking imagery with different degrees of success. Early attempts use image processing to automatically convert raster images to watercolor simulated paintings. Johan et al. [2004] focused on recreating watercolor brush strokes from photographs by following a vector field created along the boundaries of the image elements (Figure 3.4). Additional segmentation, and a set of rules for each segment, enabled some control over stroke location, interval, fluctuation and width—together with color diffusion, subject to a set of conditions and thresholds. Unfortunately, many characteristic effects of watercolors, referred to in the paper as 'pigment-based features', were not addressed, making results look like a digital painting, rather than a watercolor painting.

Figure 3.4: Johan et al. [2004].

Bousseau et al. [2006] proposed an image-space synthesis that addressed characteristic effects of watercolor (transparency, granulation, pigment turbulence and edge darkening) and abstraction, together with temporal coherence in animation. The synthesis is performed on images from either photographs or 3D rendered scenes, which are previously abstracted to obtain uniform color regions. On photographs, color abstraction is done with the help of the *mean-shift* algorithm [Comaniciu and Meer, 1999], whereas 3D scenes are abstracted using a toon shader [Lake et al., 2000] with averaged vertex normals.

Once the image has been processed and abstracted, a set of filters, divided into three layers, are used to simulate the variations found in the pigment density (caused by turbulence and granulation). First, *Perlin* noise [Perlin, 1985] is used to create a low frequency turbulent noise texture. Then higher frequency variations are incorporated through a sum of *Gaussian* noises at different scales. Finally, scanned paper textures are overlaid on the image. These three grayscale textures modify the outcome by lightening and darkening the pixel color through an empirical color modification model, based on the density and transmittance of color pigments and driven by the different gray values.

The dry-brush effect is also synthesized using the gray values offered by the scanned paper textures and masking the result according to lightness thresholds. Additionally, the wobbling (distortion) at edges is created according to the granularity of the paper, following its horizontal and vertical gradients. Finally, edge darkening is performed by using an edge detection algorithm with a symmetric kernel and is calculated within the GPU.

(a) Synthesis from image-space **(b)** Synthesis from object-space

Figure 3.5: Bousseau et al. [2006].

Spatial and temporal coherence in animation are a big problem for pure image-space approaches in non-photorealistic rendering. By using 2D generated textures for the different noises and paper texture, the approach is highly prone to the "shower door" effect. Bousseau et al. [2006] proposed two different techniques to dissipate this effect in object-space, but with limited success. A year later, they partially solved the problem in image-space, by using texture advection to make composited textures change according to the optical flow of the video [Bousseau et al., 2007]. They also introduce space-time morphological filters to coherently simplify the input animation. This approach produces relatively convincing results for noise-like textures, but it is limited in the amount of effects that it can reproduce and it requires knowledge of the full animation (before and after the frame). Nonetheless, the overall stylized outcome and synthesis was quite successful compared to previous approaches and remained the most advanced synthesis in image-space for a substantial amount of time.

Wang et al. [2014] proposed an approach that features a variety of image-space filters and computer vision adjustments (Figure 3.6). Until today, this is the most advanced watercolor synthesis developed in image-space, successfully making watercolor-specific color adjustments, automatically providing a localized level of abstraction, synthesizing effects like color bleeding and hand tremors (with gaps & overlaps as a byproduct) and implementing previous effects, such as paper granulation, pigment turbulence and edge darkening.

Figure 3.6: Wang et al. [2014].

First, to approach the color choices of watercolorists, the colors are adjusted according to a collection of real watercolor paintings. A training set of around 700 actual paintings were considered and classified in different groups according to their averages and standard deviations in *Lab* color space. The algorithm recognizes the most similar color group and suggests it to the user. Any desired group can be chosen and the colors are transferred [Reinhard et al., 2001] into the input photograph.

To automate the level of abstraction, a saliency detection algorithm [Ming-Ming et al., 2011] was implemented, which identifies the most likely regions which an artist might emphasize. A *mean-shift* segmentation follows, which abstracts the non-salient and salient regions with different kernel sizes and thresholds.

Color bleeding is synthesized by finding edges through a *Sobel* operator and randomly scattering seeds around boundaries, which satisfy a certain set of rules based on hue/intensity differences and its location relative to the salient regions of the image. These seeds are filtered with an ellipse-shaped kernel to generate an empirical color bleeding effect. Hand tremor effects, found in most traditional paintings, had not been investigated in previous watercolor approaches and are synthesized by distorting specific boundaries with *Perlin noise*. As a side effect of these tremors, gaps & overlaps are generated throughout the image. These boundaries for hand tremors are also chosen following a similar set of rules to the color bleeding effect. Edge darkening, granulation and pigment turbulence were implemented from previous methods [Bousseau et al., 2006; Lei and Chang, 2004; Luft and Deussen, 2006a]. Finally, an *FXAA* anti-aliasing filter is applied to smoothen the boundaries.

Unfortunately, the approach is strictly applied to photographs and is completely automated, so it does not offer any control. Additionally, since everything is driven in image-space, with some effects even applied randomly, this approach would not work consistently when animation is involved due to coherency problems.

In the same year as Wang et al. [2014], another image-space approach was proposed by Lü and Chen [2014]. While questionably written and arguably less successful than its predecessor, they introduce the concept of weight maps to drive a watercolor synthesis. To create this weight map, they also take advantage of saliency detection, but further combine it with edge detection to generate a weight map that helps adjust the color and abstraction of the input imagery. Once the weight map is calculated, a mixture of median and *mean-shift* filters, applied according to different weight map thresholds, smoothen the color and detail. With the color and detail smoothened, the authors begin to synthesize some characteristic effects of watercolor.

Edge diffusion (color bleeding at the edges) is synthesized by using what they refer to as a nonlinear (sic) combination of adjacent zones by linearly interpolating pixels near to the edges. Darkened edges are also generated (though not observable in the results) by subtracting the pixel value by a preferable range of 20 to 40 (sic). To reduce any resulting artifacts, generated by the previous algorithms, the authors perform morphological smoothing over the image. Finally, the authors add a paper texture that modifies the color using the color transmittance modification model proposed by Bousseau et al. [2006].

Figure 3.7: Lü and Chen [2014].

The previous image-space approaches towards watercolor synthesis are mostly automated, offering only high levels of control—if any—that will not allow skilled artists to fully express themselves. Semmo et al. [2016] proposed a framework that combines previous research, driven through multiple levels of control—default presets, global parameters, and local adjustments through a painting interface. Their work, pursued concurrently with the research presented in this thesis, also concentrates in art-directed synthesis of effects, but limited to 2D content.

3.3 Object-space Approaches

Object-space approaches, unlike stroke-space and image-space approaches, take 3D information as input to produce synthesized watercolor imagery. By having the subject in three dimensions, object-space approaches may take advantage of significantly more data from the subject and its environment. Subsequently, the gathered data can be used to help further image-space techniques towards the final watercolorized image. Therefore, object-space approaches are often performed together with image-space techniques, producing object-space approaches with a hybrid nature. The downsides from object-space approaches are that they tend to not work well solely based on images/photographs and that they require versed users with the right 3D skill-set to create the subject, in the first place.

Traditional watercolor is painted on a 2D surface, so its physical phenomena happens in 2D. Therefore, a watercolor synthesis in object-space significantly benefits from additional image-space techniques, producing more faithful characteristic watercolor effects. For this purpose, all the necessary object-space data can be gathered and prepared for image-space techniques to perform faster and better results. Curtis et al. [1997] realized this early on, by rendering their object-space subject first, and running their physical approximation on top of the rendered imagery. This was continued when Lum and Ma [2001] proposed the first dedicated object-space approach to synthesize watercolor. While they did not intend to synthesize realistic watercolor pictures, they employed watercolor techniques that an artist might use to illustrate shape and spatial relationships in 3D rendered imagery.

For this purpose, they created a pigment based lighting model, which uses a modified *Phong* shading model [Phong, 1975] to determine the thickness of each color layer—composited later in image-space using the *Kubelka-Munk* color model to approach the behavior of natural media. While there was no intention to simulate the influence of paper on the watercolor synthesis, each layer still went through a texturing phase in 3D space, which used *Perlin* noise in combination with *Line Integral Convolution* [Cabral and Leedom, 1993] along the principal curvature directions to simulate a brush-like appearance and provide variation to the layer thickness. The final outcome did not resemble watercolor imagery, but had a high degree of frame-to-frame coherence due to the noise being applied in object-space.

Lei and Chang [2004] proposed a more convincing object-space approach to synthesize watercolors, combining object-space and image-space techniques. They took advantage of programmable vertex and fragment shaders for this purpose. Their system features a *color-band specifying* approach and an image-space *Sobel* filter. The authors claim to have recreated all effects stated by Curtis et al. [1997] in real-time. However, they only synthesize a subset of them e.g., a custom watercolor

reflectance model, paper distortion, pigment granulation and darkened edges.

Color-band specifying is used to simulate a watercolor reflectance model and specific pigment granulation properties. This stage consists of manually painting with a set of predefined pigments over a separate lit-sphere interface which runs a physically-based watercolor simulation engine, based on Curtis et al.'s model. Additionally, the *Kubelka-Munk* color model is used to composite the different pigments. Once the physical simulation of the painting is done, a one-dimensional color-band is extracted from an arbitrary angle along the radius containing all the color information and granulation of the pigments used—which are stored in $RGBA$ channels.

Figure 3.8: Lei and Chang [2004].

A watercolor material is used at the second phase which consists of a vertex and a fragment shader. The vertex shader produces a color- and granulation-map from the one-dimensional color-band following a *Phong* shading function, similar to cartoon shading [Lake et al., 2000]. The fragment shader takes as input both generated maps and a paper texture and returns the watercolor styled image (Figure 3.8), as explained next.

First, the original color map is distorted by the height field (gray levels) of the paper texture to form the distorted color map C_{dst}. Then the paper texture and the granulation map are multiplied to form the paper map P_{map} and subtracted to emulate the pigment that has precipitated into the valleys of the paper. Finally, a post processing edge detection using the *Sobel* filter was implemented to create the sobel map S_{map}. The stylization is governed by: $C_o = (1 - W) + C_{dst} \times W - P_{map} \times P - S_{map} \times S$. Where C_o is the color output and the custom weights W, P, S are set by the user to accentuate the different maps (0.6, 0.1, 0.3).

Not long afterwards, Burgess et al. [2005] present a procedural combination of object-space and image-space techniques which also takes advantage of the GPU and programmable shaders for its watercolor approximation. While highly inspired by the approach of Lum and Ma [2001], the method developed by Burgess et al. approaches watercolor with real-time performance in mind, avoiding the use of *Line Integral Convolution*. However, the results do not manage to render a convincing representation of watercolors, as it only synthesizes pigment turbulence, hand tremors and edge darkening effects.

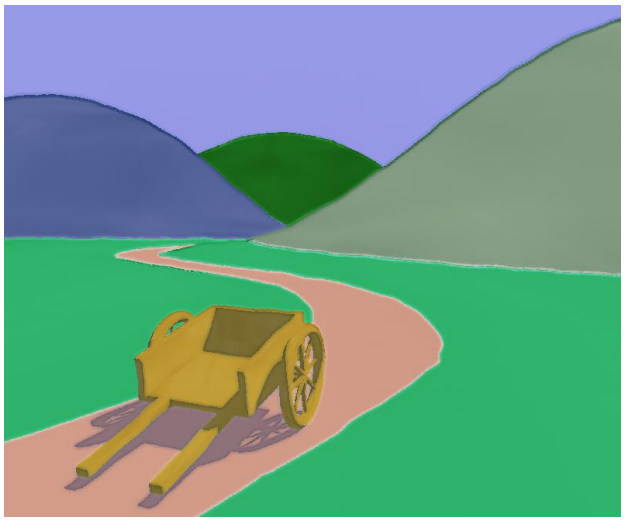


Figure 3.9: Burgess et al. [2005].

This method presented an object-space synthesis with a layered approach. A diffuse, flat colored layer is rendered, followed by a shadow layer, which is calculated using shadow volumes. Then, a texture layer is calculated to simulate pigment turbulence patterns by modifying the diffuse layer's thickness by a previously rendered *Wyvill* noise [Wyvill, 1999]. Once all layers have been processed, the shadow layer and the texture layer are saved as individual channels, together with depth information and *object ID*, inside an $RGBA$ image.

The image-space procedure first composites the texture and shadow layers, following the *Kubelka-Munk* color model. Then, edge darkening is calculated by sampling the depth and *ID* channels from 25 sparse neighboring pixels. Depending on the differences and a set of thresholds, the pigment is thickened or thinned proportionally. Finally, pixels are offset according to noise information which was not specified, emulating the imperfections of hand tremors.

Around the same time that Bousseau et al. [2006] proposed their image-space approach—that was also applied onto 3D, by stylizing objects that were rendered with a toon shader with averaged normals—Luft and Deussen started presenting their object space approaches [2005; 2006a; 2006b; 2008].

The research of Luft and Deussen initially presented a strong emphasis on synthesizing the application of watercolors in the depiction of plants, but they later generalized their approach to other objects in animation and to CAD (Computer-Aided Design) visualizations [Luft et al., 2008]. Their first three papers feature watercolor effects such as edge darkening, color bleeding, pigment granulation, paper distortion and abstraction with significant success, compared to previous methods.

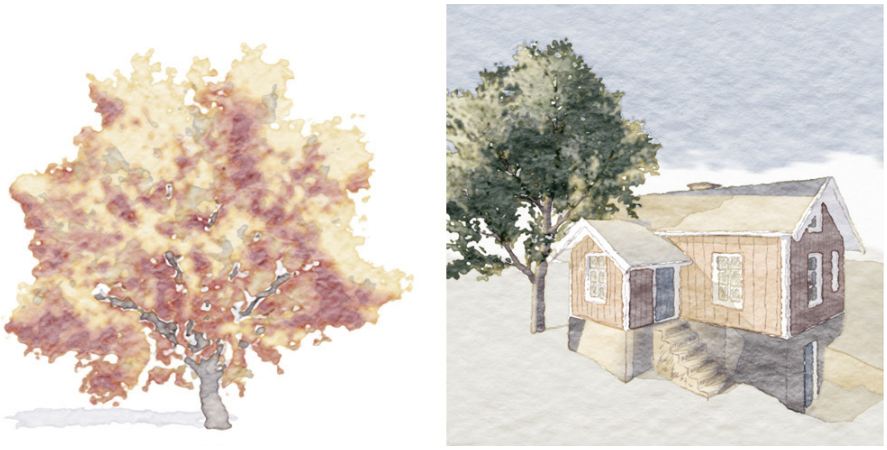


Figure 3.10: Luft and Deussen [2006b].

The system proposes a convoluted layered system, which is supposed to emulate a layered application in traditional watercolor (i.e., glazing). The semi-transparent layers proposed by the system include *IDs*, *RGBA*, specular, diffuse and shadow layers. These layers are rendered per *ID* and go through extensive processing at different points down the pipeline, as described next.

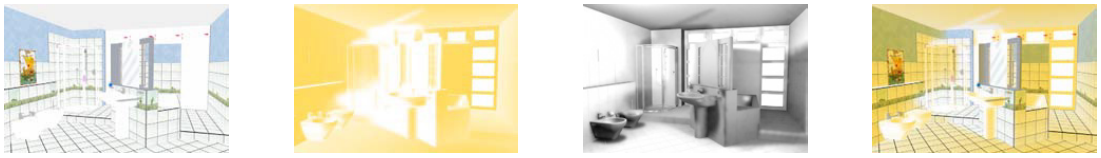
Every object or group of objects in the scene which shares a similar content/color gets an *object ID*, stored as an intensity layer—this provides object-space segmentation at minimal computational cost. A *Phong* lighting model is used, incorporating three lighting layers: specular illumination, diffuse illumination and a shadow layer through shadow mapping. The specular lighting layer is subtracted from the *ID* intensity layers as these areas remain uncolored. Additionally, the shape is changed by an intensity texture [-1, 1] representing the paper by adding the values to the *ID* layer, following a threshold. After the specular component has been removed and the shape changed by the paper structure, the *ID* layers go through a low-pass *Gaussian* filtering step, which sets the abstraction of the represented shapes. Subsequently, the smooth *ID* layers get their respective *RGBA* values assigned, whose color is specified by the user. Furthermore, to emulate wet-on-dry and wet-on-wet (color bleeding) painting techniques, a *step* and *smoothstep* functions are applied to the alpha channel.

To simulate the darkened edges, the alpha channel is then multiplied by the inverse smooth intensity values, leaving the edges of the *ID* channel opaque, while

the center becomes transparent, emulating watercolor transparency. As for the granulation, the paper intensity texture used earlier to change the shape, is used again by adding the intensity to the alpha channel. Once all these procedures have been done, the diffuse and shadow layers are incorporated. To generate multiple layers of color, the diffuse illumination layer is split through a threshold. Each of these layers is assigned a different color to emphasize a glazing technique and provide more color control. These layers also go through a low-pass filter to assimilate the painting technique applied to the image. The shadow layer goes through this filtering, as well. Subsequently, it goes through all the computation done before to recreate the watercolor effects to finally be applied into the final render. In the case of trees, further particles are also attached to the geometry to emulate leaves—this extra layers also go through the same processing as the other objects. While the resulting synthesis was, at times, successful, their approach does not scale well with scene complexity, as each group of objects (with the same *ID*) is processed individually.

Two years later, Luft et al. [2008] integrated a watercolor synthesis into a CAD (Computer-Aided Design) software called *Palette CAD* and highlighted the importance of integrating NPR into existing rendering systems. They adopt previously researched watercolor effects and contribute a tone-based lighting model and stylistic means of abstraction/indication. Their approach involves online and offline rendering to compose a stylized watercolor image out of three specific layers. The detail layer, which contains the lit scene; the ambient layer, which contains custom-colored indirect lighting; and the stroke layer, which contains object contours and cross-hatching. All these layers are previously modulated by ambient occlusion, which is rendered offline.

The paper introduces a tone-based lighting model (Figure 3.11), separating shading into tone and intensity. This is done to support custom harmonic color palettes with variations in tone and saturation, commonly used in watercolor. The approach is necessary in the system to add the light tone to the indirect lighting.



Color Light tone Light intensity Shaded image

Figure 3.11: Tone-based lighting.

The stylization takes advantage of ambient occlusion as stylistic means to denote abstraction and indication (Figure 3.12). For this purpose, they rely on the spatial geometric relationship data provided by the ambient occlusion to mask the detail and stroke layers in the areas of spatial interest. The ambient layer is modu-

lated through ambient occlusion and the light intensity to obtain areas of indirect lighting to be recolored, as desired.

Ambient Occl. Detail layer Ambient layer Stroke layer

Figure 3.12: Ambient Occlusion as stylistic means.

The visual results do reproduce a natural-looking synthesis of watercolor illustrations with soft lighting—often found in architectural visualizations (Figure 3.13). However, as a watercolor simulation system, it is not versatile and is limited to a specific look. While this is the latest work dedicated to object-space synthesis of watercolors before our contributions, there are a few other papers relevant to our work, which are surveyed next.

Figure 3.13: Rendered watercolor illustration by Luft et al. [2008].

3.4 Additional Literature

There is other work worth mentioning in this thesis. While some of the following literature might not be directly relevant to our approach, they do contribute to the overall body of work towards the synthesis of watercolors.

A curious use and approach towards synthesizing watercolors has been developed in the field of augmented reality [Chen et al., 2008]. To avoid the computation and incorporation of photoreal 3D renders over interactive video, Chen et al. decided to stylize everything, instead. Simple, 3D rendered elements are composited into every video frame, where an image-space procedural approach would synthesize a watercolor-inspired look. Their approach makes use of *Voronoi* diagrams, which are tiled to follow the edge information of each frame. While it does not successfully synthesize a watercolor look, it fulfills the purpose of making the physical world and the virtual world indistinguishable from each other (Figure 3.14).

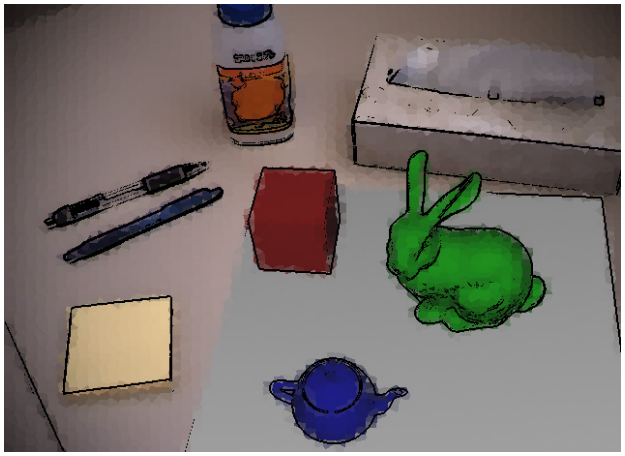


Figure 3.14: Chen et al. [2008].

The use of image analogies in example-based approaches is also capable of synthesizing a watercolor look in object-space. In the approach presented by Bénard et al. [2013], an artist paints a set of frames directly onto a 3D model. If a stroke-space synthesis is applied at this stage, the system will try to interpolate the painted frames—which are projected onto the 3D geometry. This approach can support a wide range of painterly styles and grants extensive artistic control over them (Figure 3.15), but requires the artist to paint a new frame every 10 to 20 images on average. Therefore, it is not feasible in interactive real-time 3D graphics, which require complete freedom of view and movement.

Fišer et al. proposed two other example-based approaches that are capable of supporting a watercolor-look. One approach (Figure 3.16a) simulates noise from static watercolor textures placed in image-space and object-space (mapped to an object), forcing incoherences that would normally happen when painted by hand

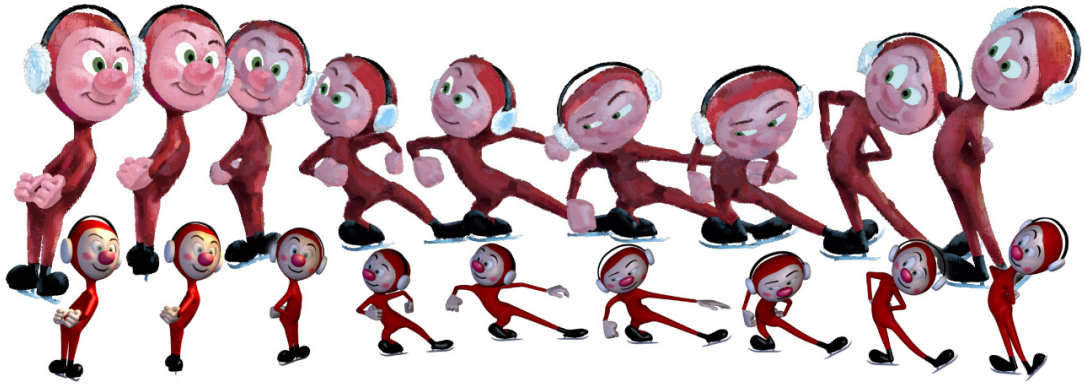


Figure 3.15: Bénard et al. [2013].

[Fišer et al., 2014]. The other example-based approach (Figures 3.16b and 3.16c) acquires a traditionally drawn lit sphere and synthesizes the 3D objects' context-dependent illumination effects based on light propagation [Fišer et al., 2016]. While these example-based approaches do not require the user to paint multiple frames for them to work under animation, they are not applicable in real-time, lack intuitive local control of the style transfer and present their own additional caveats, which make them impractical for current production pipelines.

Finally, another approach worth mentioning is the synthesis of watercolors through style transfer using convolutional neural networks [Gatys et al., 2016]. While extremely simple to synthesize for the end user, the results of neural networks cannot be art-directed and output incoherent results under animation—making this approach unsuitable for serious use.

Considering all the prior research, one could argue that the synthesis of watercolors has already been partially solved using offline approaches. Therefore, my work concentrated in taking and improving existing research, optimizing it for real-time application and expanding it towards additional effects and a more faithful

(a) Synthesized noise **(b)** Painted lit sphere **(c)** Synthesized style transfer

Figure 3.16: Consolidated work of Fišer et al.

synthesis. In addition to this, I put a strong emphasis on enhancing control over the synthesis, complementing the new stylization approaches with intuitive ways to art-direct the outcome. From the next chapter, the contribution of my studies will be presented.

4. Painterly Material

Materials give the surface properties to 3D objects, which are then projected to an image. At their essence, they consist of a set of shaders—which are algorithms that are run at different stages of the graphics pipeline—and oftentimes the definition of a user interface to interact with the shader parameters. Since most of the computer graphics industry has been on a quest towards photorealism since its inception [Durand, 2002; Salesin, 2002], a lot of research has focused on creating and controlling photorealistic materials [Schmidt et al., 2016]. However, since watercolors are not commonly used to depict photorealistic subjects, different types of materials are required, which better reflect the usage and workflow of the natural medium. Non-photorealistic materials have been proposed for some time [Gooch et al., 1998; Lake et al., 2000; Mitchell et al., 2007; Todo et al., 2013; Vanderhaeghe et al., 2011], but these have lacked some painterly properties commonly found in watercolors e.g., pigment dilution, cangiante reflectance. Therefore, previous work synthesizing watercolors has also proposed the use of custom materials [Bousseau et al., 2006; Lei and Chang, 2004], but these present caveats, as well.

Since real-time interaction is one of the fundaments of this research, I propose a painterly material that runs on the *graphics processing unit* (GPU). The material focuses on bringing colors into the foreground of shading control, but also on being flexible enough to support common photorealistic shading techniques such as texture/normal mapping, specularity and transparency—to avoid disruption of existing workflows. However, most importantly, the material needs to be able to support watercolor effects.

Before going in depth into technicalities, I would like to introduce real-time rendering for the non-technical reader. Real-time graphics are normally calculated using a *graphics processing unit* (GPU), instead of a *central processing unit* (CPU), which is used for most computing tasks in a computer. A powerful GPU nowadays offers a massive amount of processors, which can be used to perform calculations in parallel. While each one of these processors is generally slower than the CPU found on each computer, there are several applications that can take advantage of the parallel nature of GPUs, exponentially improving the speed at which something is computed. Many of these applications can be found in computer graphics and are

addressed by programs called shaders. In simple terms, a shader is a program which runs in each parallel processor. There are several different shader stages involved when rendering 3D graphics, however, we are mostly concerned with the *vertex shader* and the *pixel/fragment shader* stages of the graphics pipeline (marked in teal color in Figure 4.1). In the following, I will only introduce the relevant parts of the graphics pipeline, but please refer to *GPU Performance for Game Artists* [O'Conor, 2017] or to some of the established technical literature on the topic [Akenine-Moller et al., 2018; Kessenich et al., 2016] for in-depth information.

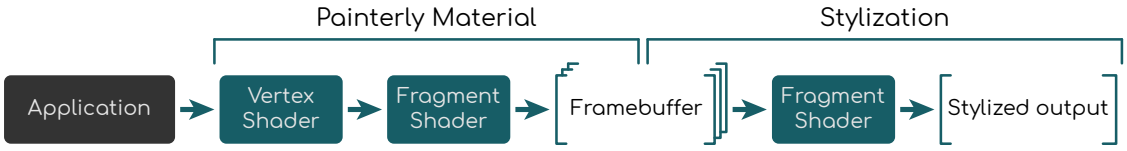


Figure 4.1: Simplified GPU stylization pipeline.

The *Application* contains all the raw geometric information that needs to be rendered. For this purpose, the geometric information is sent to the *Vertex Shader* stage, which proceeds to do calculations over the vertices and transforms them for future projection to an image plane. The first *Fragment Shader* stage takes the rasterized, projected object-space data and calculates each pixel color based on the material properties, lighting and environment information. The pixels are saved into the *Framebuffer* as *render targets* (images). The second *Fragment Shader* takes these images and performs further calculations over the pixels, to synthesize different effects required for the *Stylized Output*.

The material is the core of an object-space stylization approach and is in charge of the first two shader stages. It receives the geometric data from the application to produce the different render targets stored in the framebuffer. This chapter will break down the research and contributions towards a painterly watercolor material.

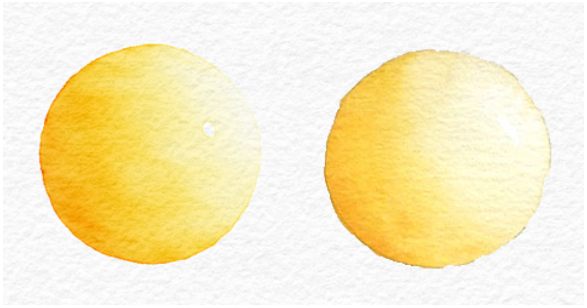


Figure 4.2: Our watercolor synthesis (left) and hand-painted (right).

4.1 Reflectance Models

Photorealistic rendering revolves around the interaction between light and matter [Pharr et al., 2016], whereas painting revolves around applying perceived and interpreted colors. This is a significantly different approach towards creating imagery, which needs to be addressed accordingly. Therefore, various non-photorealistic reflectance models have been proposed by previous researchers [Gooch et al., 1998; Lake et al., 2000; Mitchell et al., 2007; Todo et al., 2013; Vanderhaeghe et al., 2011]. A reflectance model dictates how light (color) is reflected along the surface of an object. The characteristic translucency found in watercolors plays a pivotal role in the way motifs are colored, but is not found in shading primitives used by common reflectance models.

Lei and Chang [2004] proposed a material that uses a one-dimensional color ramp to define reflected colors throughout the surface. This can prove useful and highly customizable as a color can be set to reflect at each specified angle. However, this approach offers disadvantages over other empirical reflectance models used for real-time graphics. Colors might end up in undesired places, as they are solely based on the surface angle in relation to the light source. Each different section of a one-dimensional color-band needs to be set for each material and changed whenever a different color is desired. More importantly, this approach conflicts with the use of texture maps, as color is defined solely by the color ramp and geometric details.

Bousseau et al. [2006] proposed the use of cartoon shading [Lake et al., 2000], driven by averaged normals. In this approach, the light intensity (instead of color) is specified at different angles, allowing the use of texture maps. However, the light sections at specified thresholds might propagate in undesirable ways and the fixed aesthetic is limiting for artists looking for a more organic depiction.

Luft and Deussen [2006a] do not propose the use of a custom material, but they introduce a technique to synthesize the characteristic transparency of watercolors by modulating the alpha channel of colored regions. However, relying solely on alpha transparency does not represent the way a watercolor reflectance model should ideally work. This may seem unintuitive and difficult to assimilate for the untrained eye, but artists do not darken and lighten a painting using black and white.

> *True highlights (not reflections) on a color do not necessarily shift towards white, and true shadows on a color do not necessarily shift toward black.* (Faber Birren [1987])

According to Birren [1987], this phenomena and its application can be traced back to Da Vinci and his *chiaroscuro* style which revolutionized the art of painting. Referred by Edward Hering as a *veiled sequence* and by Wilhelm Ostwald as *shadow series*, this effect can also be observed in photography. Eventually, the highlight will perceptually be white at some point if the intensity of the light is strong enough. However, it goes through a change of hue first to a lighter, related color.

Shading using black and white will make the painting look muddy and dull. A clear example is given in Figure 4.3, where white and yellow do not reproduce the color of the highlights and black rather turns the yellow color towards green. This happens because, when different amounts of light reflect from a surface, the hue changes towards related colors, as well [Parramón's Editorial Team, 2000].

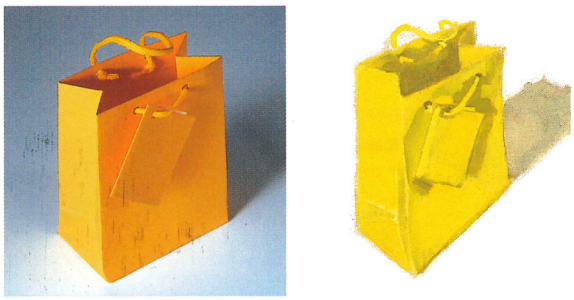


Figure 4.3: A painting lit using black and white, Parramón's Editorial Team [2000].

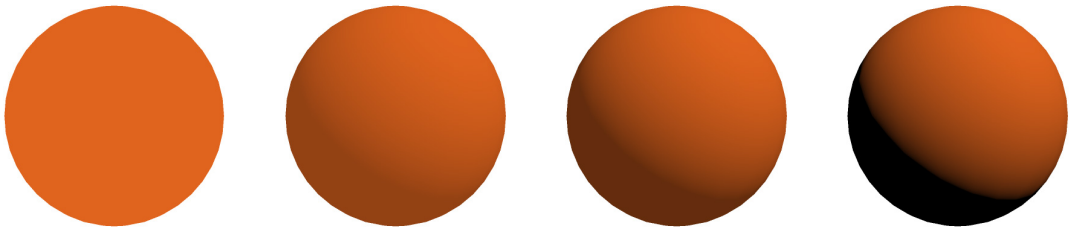
To overcome these limitations and address the characteristic transparent and painterly color application found in watercolors, we propose a custom painterly reflectance model. Our shading model features a controllable diffusion factor, which enables the depiction of flat and shaded objects, and the ability to control color on lit and shaded parts of objects, while supporting photorealistic texture mapping techniques.

4.1.1 Diffuse Control

Depending on the intended depiction, an artist may choose to portray 3D objects in a flat or diffuse manner. Diffuse reflectance can be empirically approximated by taking the dot product of the surface normal and the light vector i.e., $\vec{N} \cdot \vec{L}$. This approximation is commonly known as *Lambertian* reflectance and was introduced by Johann Heinrich Lambert [1760]. While not accurate enough for photorealistic depiction, this approximation is fast, simple and predictable for artists to use. By using an additional diffuse factor d_f, the dot product $(\vec{N} \cdot \vec{L})$ can be linearly interpolated between a flat-shaded object and a diffuse object (Equation 4.1). This

allows for a custom diffuse contribution D that, when multiplied by the material color C, produces the diffuse color C_D (Figure 4.4). The diffuse factor can also empirically approximate a global illumination per object, without the artist having to add additional lights or otherwise use ray-tracing.

$$D = d_f(\vec{N} \cdot \vec{L} - 1) + 1 \tag{4.1}$$

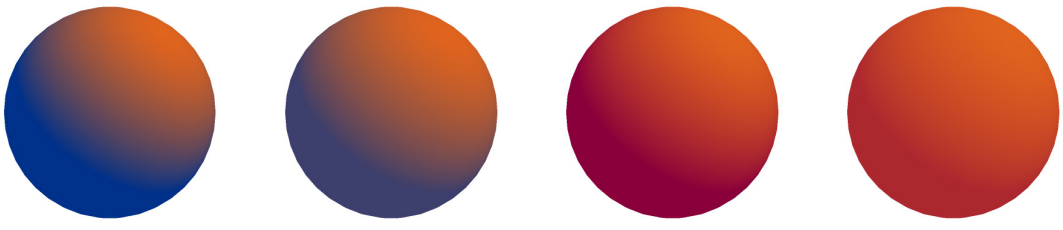


(a) $d_f = 0$ **(b)** $d_f = 0.33$ **(c)** $d_f = 0.66$ **(d)** $d_f = 1.0$

Figure 4.4: Diffuse colored sphere, $C_D = C \times D$.

An artist does not usually use black when shading their subjects, as they have the creative freedom to apply color as they desire. Therefore, apart from using the modulated diffuse reflectance to multiply it by the color (Figure 4.4), we propose a way to override the color in shaded areas, if desired—similarly to Gooch et al. [1998]. Using the modulated diffuse contribution D, we can linearly interpolate an arbitrary shade color C_s instead, generating a diffuse color override C_{Do} (Equation 4.2) as illustrated in Figure 4.5.

$$C_{Do} = D(C - C_s) + C_s \tag{4.2}$$



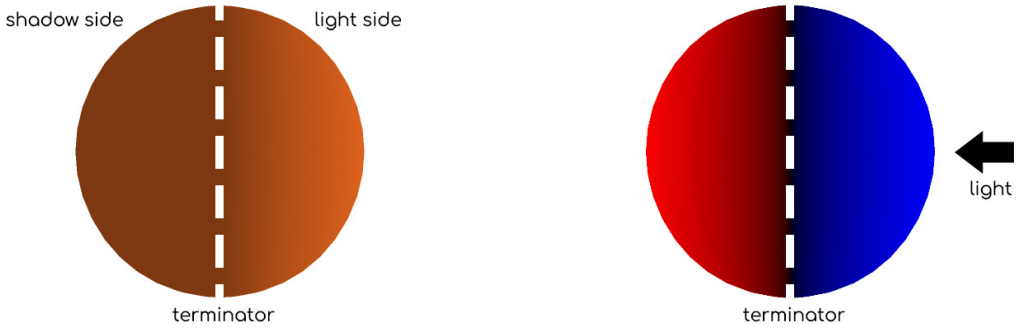
(a) $d_f = 1.0$, $C_s = (0, 0.192, 0.54)$ **(b)** $d_f = 0.5$, $C_s = (0, 0.192, 0.54)$ **(c)** $d_f = 1.0$, $C_s = (0.54, 0, 0.231)$ **(d)** $d_f = 0.33$, $C_s = (0.54, 0, 0.231)$

Figure 4.5: Examples of diffuse shading with shade color override.

Overriding the shading color is useful, but often a more defined color control is required, especially on the lit and shaded sides of objects.

4.1.2 Shade Control

A *terminator* is the delimiter between the light and shadow side of an object, and the dot product of the surface normal and light direction provides a useful mathematical separation between lit and shaded areas (Figure 4.6b). We can take advantage of this to provide additional artistic control.



(a) Shadow side with a solid color.

(b) $\vec{N} \cdot \vec{L}$, blue is positive, red is negative.

Figure 4.6: Visualizing the terminator delimiter.

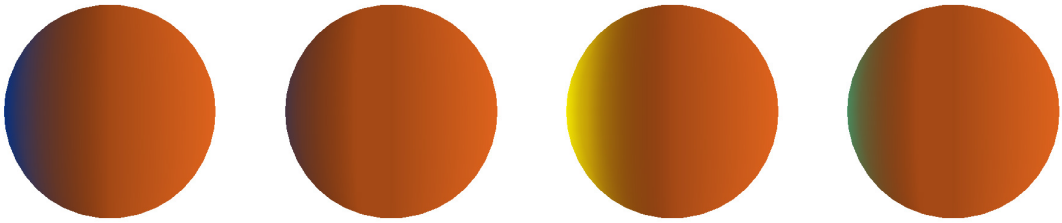
When we use our empirical diffuse approximation, the shadow side of objects has a solid shade color (Figure 4.6a), which is not commonly desired. In photorealistic approaches, bounced lights give the shade intricate additional light information, based on its surrounding environment. An artist approximates this by adding color to the shadow side, either based on the surrounding elements or their artistic interpretation.

To enable a similar aesthetic, we first take the negative part of the *Lambertian* reflectance (red in Figure 4.6b) and modulate it by a shade wrap s_w parameter to obtain a custom adjustable shade area $S_A \in [0, 1]$ (Equation 4.3).

$$S_A = \frac{(s_w - 1) - \vec{N} \cdot \vec{L}}{s_w} \tag{4.3}$$

The custom shade area S_A is then used to linearly interpolate any desired shade color C_s into the diffuse approximation C_D, creating the output color C_o (Equation 4.4). The interpolation wraps the shadow side with the shade color C_s covering the shade area S_A.

$$C_o = S_A(C_s - C_D) + C_D \tag{4.4}$$



(a) $s_w = 1.0$
$C_s = (0, 0.192, 0.54)$

(b) $s_w = 0.7$
$C_s = (0, 0.192, 0.54)$

(c) $s_w = 1.0$
$C_s = (1.0, 0.964, 0)$

(d) $s_w = 0.7$
$C_s = (0, 1.0, 0.794)$

Figure 4.7: Examples of shade wrap being applied to the shadow side of objects.

4.1.3 Light Control

The cangiante illumination (change in the highlights to a brighter hue) and characteristic transparency found in watercolors play a pivotal role in the way motifs are colored, but it is not found in common shading primitives. While the previously introduced equations govern the shading of objects—with the possibility to customize the shade of a lit object—artistic control can also be given to the light side of objects (Figure 4.6). Painters resort to brighter hues when painting highlights and watercolorists make additional use of the transparent property of traditional watercolors to show any color underneath. The transparency of watercolors is often used to show lit areas, by allowing the paper color to shine through. Therefore, the white of the paper is often used to lighten the depicted objects, by diluting transparent watercolors.

To synthesize cangiante illumination and dilution, we first need to calculate the area of effect for each, in a similar manner to the shade wrap area, by modulating the dot product of the surface normal $\vec{N}$ and the light direction $\vec{L}$, with a cangiante c_w or dilute d_w wrap parameter.

$$C_A = \frac{(c_w - 1) + \vec{N} \cdot \vec{L}}{c_w} \tag{4.5}$$

$$D_A = \frac{(d_w - 1) + \vec{N} \cdot \vec{L}}{d_w} \tag{4.6}$$

With the computed wrap areas, the surface color may undergo the changes of a cangiante illumination, shifting the hue towards a nearby primary/secondary 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 [Birren, 1987]. To

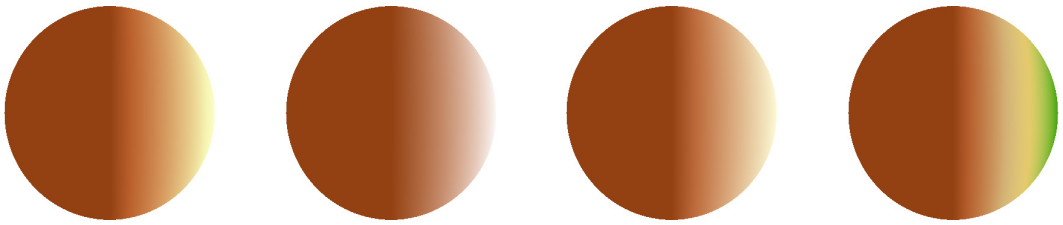
achieve this, the cangiante color C_c is calculated by adding the cangiante wrap area C_A, multiplied by the cangiante variable c to the surface color C (Equation 4.7).

$$C_c = C + (C_A \times c) \tag{4.7}$$

Once the hue at the highlight has shifted, the dilution is performed by linearly interpolating the surface color towards the paper color C_p, through the dilution variable d (Equation 4.8).

$$C_d = d \times D_A(C_p - C_c) + C_c \tag{4.8}$$

The combination of cangiante illumination and dilution allows for a wide spectrum of control on the light side of objects (Figure 4.8). Additionally, by setting arbitrary colors as paper color C_p in the material, a similar effect to shade wrap can be achieved, but on the light side of the material, instead. This allows any color to be wrapped onto the light side of objects (Figure 4.8d).



(a) $c_w = 1.0$
$c = 0.6$

(b) $d_w = 1.0$
$d = 1.0$
$C_p = (1.0, 1.0, 1.0)$

(c) $c_w = d_w = 1.0$
$c = 0.5, d = 0.5$
$C_p = (1.0, 1.0, 1.0)$

(d) $c_w = 1.0, d_w = 0.56$
$c = d = 1.0$
$C_p = (0, 0.185, 0.608)$

Figure 4.8: Examples of cangiante illumination and dilution applied with different parameters on the same material.

When used in combination, the dilute and cangiante behavior of the painterly material enable the synthesis of the characteristic *transparent shading* found in traditional watercolors (Figure 4.2).

4.1.4 Additional Control

The proposed painterly material provides extensive control over the diffuse shading, but shiny objects require an additional specular reflectance, as well. To support

shiny materials, the material includes the *Blinn* isotropic [Blinn, 1977] and *Ward* anisotropic [Ward, 1992] specular reflectance models. Both of these reflectance models are complemented by *roll off* attributes that define the size of the specularity, a *diffusion* attribute that defines the transition of the specular highlight and a *transparency* attribute to control the intensity of the specularity.

The painterly material offers vast control over color, but common workflows do not rely solely on colors set within material parameters. Most workflows rely on textures, mapped to the objects themselves. Naturally, to motivate the usage of our material in existing workflows and productions, different texture maps are supported, whose color can further be tinted with the same color control, as elaborated in the previous sections. The supported texture maps comprise the following:

- *Albedo*. The albedo texture contains the raw colors of the object, without any lighting information.
- *Normal*. The normal texture contains the normal inclination deviations in the red and green channels. This texture is used to add geometric detail through normal modulations [Cignoni et al., 1998; Cohen et al., 1998]. An additional *bump depth* attribute controls the intensity of the normal inclinations.
- *Specular*. The specular texture contains the weight map for the specular reflections. It maps and defines the specular areas within an object.
- *Alpha*. The alpha texture contains the weight map for the transparency in different parts of the object.

All these textures are uv-mapped by the painterly material to the objects. The resulting material proposes a versatile workflow for both, the artist accustomed to thinking about colors and their application, and the artist accustomed to creating texture maps with more traditional 3D workflows (Figure 4.9).

While the painterly material offers control over the physical appearance, it is only able to synthesize the characteristic transparent shading found in watercolors and general cangiante illumination. The other characteristic effects remain to be solved in our approach. However, the synthesis of these effects is better suited in image-space, as the physical phenomena and behavior of watercolors happens on a 2D plane (paper).

(a) Fruit plate model.

(b) Utah teapot model.

(c) Gotita model.

(d) Spherebot model, CC BY Bastien Genbrugge.

(e) Wiz Biz model, CC BY Tom Robinson.

(f) Baba Yaga's hut model, CC BY Inuciian.

Figure 4.9: Painterly materials applied to objects/scenes, stylized in Figure 5.23.

4.2 Effect Drivers

Most watercolor effects in our approach are synthesized in image-space, but image-space approaches rely on image processing, which is usually performed over all the pixels of an input image. Artists do not work like this, as they will not paint their entire artwork with an even treatment. They will emphasize, abstract, and add custom effects locally, following 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. To control and drive these effects, low-level (local) control over the stylization is required.

Masking has been commonly used for image-space manipulation and compositing, but recent image-space stylization systems have also began adopting them to drive the synthesis of effects. To this end, painting effect parameter masks has been proposed, to offer low-level control over effect synthesis in image-space [Semmo et al., 2016]. These parameter masks work well for single images, but the masked effects do not follow any of the underlying objects under animation, remaining fixed on the screen. This contributes to the often encountered and termed "shower door" effect in non-photorealistic rendering, as the effects are not spatially coherent in relation to the subjects [Bénard et al., 2011]. To address this problem and make the assignment of effects spatially coherent, effects can be applied directly to the geometry. However, as we are not synthesizing any other effects in object-space, this imposes an exciting paradox. Controls should be embedded in object-space, so as to warrant spatial coherence, but we are not synthesizing these effects in object-space.

The key lies in the word "embedding", as we can assign control parameters in object-space and render these previously assigned parameters into control masks for future image-space stylization control. The object-space data then drives the effects in image-space, remaining spatially coherent and taking advantage of image processing algorithms for the effect synthesis. Our masks are not assigned by painting pixels on a 2D plane, but by painting vertices (points) in 3D space, that are rasterized into pixel masks. Alex Harvill [2007] has also implemented this idea through an offline stylization pipeline, rendering a mask of weighted points to control a moving 2D illustration. However, in addition to painting vertices, we also provide procedural noises within the material that are mapped onto the 3D objects. This higher level of control can give the painted effects some random variation, augmenting the unpredictable qualities of watercolor.

The object-space control parameters can be assigned in vertex colors, point weights, noise maps or custom texture maps. Any representation is possible, as long as the scalar control data can be used to render parameter masks for further

image-space stylization. In our approach, we have chosen the use of vertex colors, as they efficiently provide discretized parameters within four channels (RGBA). Furthermore, they are natively supported by vertex shaders (compute fast), can be animated for further temporal control and are supported by most 3D applications. The effect parameters and their influence over the stylization can be observed in Figure 4.10. The main caveat of vertex colors is that they are dependent on the topology of the geometry, so more detailed control is achieved by adding control points/vertices, which requires changes to the model. Further noise maps are generated using 2D or 3D *Simplex* noise [Perlin, 2002], provided by our development framework.

In summary, the proposed painterly material is in charge of rendering all the required object-space data for future stylization, which, as of our latest implementation, includes: Z-depth, surface colors, lighting contribution, specular contribution and effect control parameters.

The data is rendered into multiple render targets, which are stored in the *framebuffer* (Figure 4.1). This can be efficiently done without re-rendering the scene, by taking advantage of the *fragment shader*'s ability to render to multiple render targets. These render targets can then drive the synthesis in image-space to produce the final stylization (Figure 5.23).

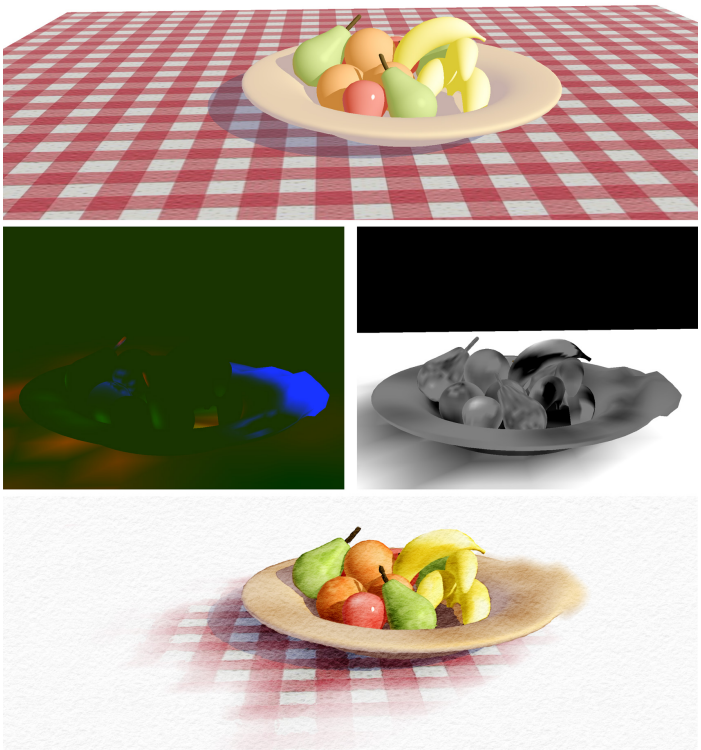


Figure 4.10: Top to bottom: color image rendered without stylization; control image showing the art-directed effect control parameters [vertex colors] (Left: RGB channels | Right: Alpha channel); art-directed watercolor synthesis.

4.3 Discussion

The painterly material introduced in this chapter saw various iterations before reaching its refined state. Initially, I pushed part of the watercolor effects synthesis towards object-space within the painterly material, by mapping the paper texture into the object, deforming the objects at the *vertex shader* stage along the normals and by modifying the surface color to simulate pigment turbulence (Section 5.1.1). These attempts, while partially successful, presented a couple of caveats.

The initial idea of mapping the paper texture onto each object was soon abandoned, as the texture would present geometric and perspective distortions—not to mention that areas without objects would not present any paper texture, which is not the case within watercolor paintings. An image-space placement of the substrate (paper) texture made more sense, particularly for still images—more on this in Section 5.3.

Using the *vertex shader* stage (Figure 4.1) to deform the objects containing the painterly material was relatively sound. One could leverage the parallel nature of shaders to easily deform objects, pulling out vertices when color bleeding was painted, or jittering them to emulate hand tremors. While this worked well with shaded objects, it presented overheads when incorporating shadow maps [Williams, 1978]. Shadow mapping re-renders the scene to produce depth maps from the light's perspective, whose distance is later compared at visible pixels from the viewer's perspective to determine if the current pixel is within an occluded surface or not. This meant that any shader-based deformations had to be reproduced in these renders, as well—creating additional vertex calculations. These extra calculations can be avoided, considering that a mesh deformer within the 3D application could do this once, and then pass the geometry to the rendering pipeline.

Creating pigment turbulence in object-space worked well for the watercolor synthesis, providing mapped control at the vertex level. However, once the developed framework was extrapolated to other styles (Section 6.2), this effect was moved to be applied in image-space through the stylization pipeline.

As my work evolved to support multiple stylizations at later stages of my PhD studies, the painterly material was left to be more generic. This meant that the refined material, exposed in this chapter, was to be only concerned with the application of color and the rendering of object-space data (e.g., depth, effect parameters)—leaving the synthesis of effects to the stylization pipeline in image-space. However, as the material was developed using the *ShaderFX* node framework provided by *Autodesk® Maya®*, it remains highly flexible and customizable by visual development artists.

Once all the object-space data has been rendered by the painterly material into the different render targets in the *framebuffer*, the stylization pipeline takes over the synthesis of the effects and the watercolor style, which is introduced next.

5. Watercolor Stylization

The watercolor stylization is responsible for taking the 3D rendered data, stored in the *framebuffer*, and synthesizing the watercolor effects, generating a stylized output (schematic, Figure 4.1). At its essence, it consists of a set of image-space *fragment shaders* that process images down a pipeline: the stylization pipeline (Chapter 6).

Different watercolor stylizations have been proposed throughout the years in image- and object-space approaches, each concentrating on synthesizing various characteristic effects of watercolor (Chapter 3). The contribution of my research lies in expanding the existing synthesis with novel algorithms and improving upon the palette of synthesized watercolor effects, generating a new state-of-the-art watercolor stylization in object-space, which is synthesized in real-time.

Based on our research, observations and discussions working with artists, the characteristic effects of watercolors can be categorized into four distinct groups:

- Pigment-based effects (Section 5.1)
- Edge-based effects (Section 5.2)
- Substrate-based effects (Section 5.3)
- Abstraction-based effects (Section 5.4)

These groups are deconstructed in this chapter, introducing the methods and algorithms to synthesize the properties and various essential characteristic effects of watercolors (Section 2.1.2).

Figure 5.1: Our watercolor stylization. Henry model, © Oculus.

5.1 Pigment-based effects

The way in which pigment is applied onto and settled on the substrate (i.e., paper, canvas, wall) largely defines the traditional painting medium. Therefore, pigment-based effects represent the core of the watercolor stylization. Watercolor is a versatile painting medium that can be applied in diluted or viscous states, presenting different pigmentation properties and effects.

When watercolors are applied in a diluted and fluid state, its characteristic transparency and dispersion allow pigments to produce various visual effects. Pigments can disperse and settle unevenly, letting the transparent property of watercolors generate pigment turbulence (Section 5.1.1), creating areas of varying pigment density and color concentration. Gravity will also disperse and pull the pigments of a wet application towards the valleys of a rough substrate (Section 5.1.2), generating a higher color concentration and featuring the paper profile—commonly referred to as paper granulation.

When watercolors are applied in a viscous and concentrated state, they behave similarly to its oil or acrylic counterpart. Applied with light pressure over a rough substrate, the pigmentation will only stay at the peaks of the paper, generating a dry-brush effect (Section 5.1.2). The application of pigments is highly influenced by the substrate, so we take advantage of accurately acquired substrates for a more faithful synthesis of the medium (Section 5.3).

To synthesize these characteristic pigment-based effects, we investigate the physical causes and any previous approaches that have addressed its synthesis. Then, based on this background, I propose and introduce our real-time approach.

5.1.1 Pigment Turbulence

When pigments settle unevenly on the flat surface of the paper, they create areas of varying pigment density, producing a low-frequency noise known as pigment turbulence. This effect is characteristic for watercolors, due to the fluidity of its carrier: water.

Burgess et al. [2005] most likely proposed the first approach towards pigment turbulence, by extending the work of Lum and Ma [2001] that used Line-integral Convolution (LIC) [Cabral and Leedom, 1993] to emulate strokes along the ob-

ject's surface. Instead of strokes, Burgess et al. [2005] aimed to generate textured thickness (pigment turbulence), by distorting *Wyvill noise* [Wyvill, 1999] along the surface of the geometry. The resulting thickness would then modify the color using the *Kubelka Munk* color model [Haase and Meyer, 1992]. Unfortunately, pigment turbulence does not necessarily follow the geometry of the subject, which resulted in synthetic patterns that did not convincingly reproduce the effect.

Bousseau et al. [2006], proposed to have noise with two different frequencies overlaid across the entire image. The noise would then modify the pixel color using a custom color transmittance modification model. The synthesis of pigment turbulence using image-space noises is convincing for still images and was also similarly adopted by Wang et al. [2014]. However, noise should not simply be overlaid over the entire image, as doing this would not distinguish between different painted areas—which would not share the same turbulence. Additionally, the noise texture would remain static under animation as a "shower-door" effect.

To overcome previous limitations and provide extended control over the pigment turbulence, we let the user art-direct the pigment density variations in object-space to suit the desired amount/pattern. Additionally we provide object-space noise to add fluctuations, incorporating a bit of randomness into the stylized outcome. The combination is achieved by letting the user directly paint the density parameters d_{fx} on objects within the 3D application, and then offset these density parameters with noise values within the painterly material in object-space. In the case of pigment turbulence, procedural density parameters can significantly accelerate the assignment of initial density variations, but art-direction is required to refine the result (Figure 5.2).

With the density parameters assigned, the accumulation of pigments is synthesized by altering the source colors through the color transmittance modification model, introduced by Bousseau et al. [2006]. 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 density parameters to fade the rendered color C towards the substrate color C_s (paper).

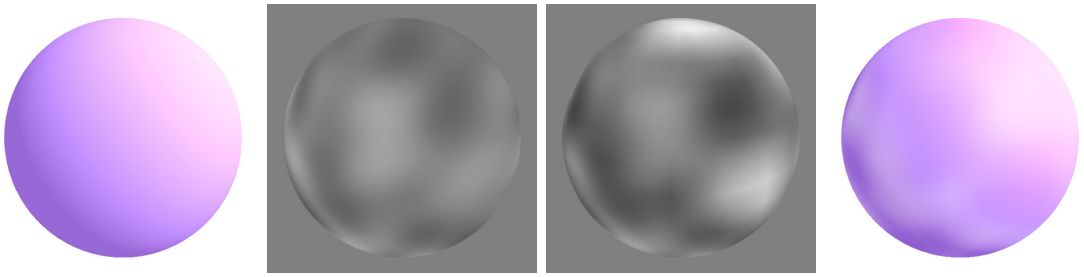


Figure 5.2: Pigment turbulence breakdown, $C = (0.66, 0.45, 0.92)$.

This is a common stylistic choice found in watercolor illustrations to direct the attention of the viewer towards a desired subject. Thereby avoiding unnecessary scene elements by diluting the color towards the paper. Both algorithms governing the pigment turbulence P_t effect are elaborated in Equation 5.1.

$$P_t = \begin{cases} C^{d_{fx}+1} & \text{if } d_{fx} \geq 0 \\ (d_{fx}+1)(C^{d_{fx}+1} - C_s) + C_s & \text{if } d_{fx} < 0 \end{cases} \tag{5.1}$$

5.1.2 Granulation and Dry-brush

Depending on how the user applies watercolor pigments over the substrate, two effects can be produced, granulation and dry-brush. When pigments are applied in a fluid state, gravity will pull the pigments towards the valleys of a rough substrate, concentrating the color and creating the paper granulation effect (Figure 5.3a). When pigments are applied lightly in a viscous ('dry') state, pigments will only settle at the peaks of a rough substrate, creating the dry-brush effect (Figure 5.3b). These effects could also be considered substrate-based effects, as they heavily rely on the substrate data. However, these effects only appear if the pigments are applied in a specific manner, qualifying them as pigment-based effects.

(a) Pigments accumulating in the valleys of the paper. **(b)** Pigments settling only at the peaks of the paper.

Figure 5.3: Visualizing the granulation and dry-brush effects.

Curtis et al. [1997] first addressed the synthesis of these effects in their physically-based approximation. The accumulation of pigments in the valleys of the paper and the dry-brush effect threshold were specified prior to running the fluid simulation over a procedurally synthesized paper. While achieving believable results, running a fluid simulation for this purpose is not feasible for real-time synthesis.

Lei and Chang [2004] proposed a procedural approach towards paper granulation that subtracted the paper texture intensity from the color. The paper texture intensity was previously modulated by a granulation parameter, extracted from a one-dimensional granulation map that was acquired from a physically-based approximation on a lit-sphere model. Unfortunately, subtracting the intensity of a

scanned paper pushes the color towards black, it does not concentrate the color.

Luft and Deussen [2006a] proposed to synthesize granulation by subtracting the paper texture intensity from the alpha channel. This would lead to transparent (brighter) colors instead of concentrated colors, which is not the right behavior.

Bousseau et al. [2006] got over these limitations, synthesizing granulation by taking the intensity of a scanned paper texture and adding it to the density parameter of their color transmittance modification model. This resulted in a concentrated color and a more believable granulation effect, which was also included by Wang et al. [2014] in their image-space approach. Bousseau et al. also proposed to use the scanned paper intensity to threshold the color and synthesize the dry-brush effect. Their granulation approach showed promising results, however, the scanned paper data did not allow for a successful dry-brush synthesis and was mainly omitted from their results.

It is quite unfortunate that the synthesis of the dry-brush effect has not been properly addressed by previous empirical approaches, even though it is closely related to granulation. The dry-brush application of pigments is a technique that is often used in watercolor painting [Franks, 1988], but it also finds wide use beyond watercolor, e.g., in comic-book inking [Janson, 2003, 15, 104, 107] and fashion illustration [Morris, 2010, 61]. An excellent use of a dry-brush application in watercolors is demonstrated in the 2011 painting by Joe Cartwright, depicted in Figure 5.5a.

Dry-brush is a painting technique that is applied by watercolor artists with the purpose of indicating a textured appearance. Its uses vary substantially from artist to artist, but it is often employed to depict rough textures, such as those produced by leaves, clouds or reflections. The rough appearance is caused when viscous ('dry') pigment only reaches the peaks of the substrate. This allows its valleys to remain unpainted, showing their color (or previously painted pigment). As a consequence, the appearance of the dry-brush application varies strongly depending on the substrate profile, the amount of pigments deposited and the pressure, direction and speed at which the brush-stroke is placed. Our approach will not consider the strokes themselves, but rather the resulting general appearance of a dry-brushed area.

For a successful synthesis of dry-brush and granulation, an accurate substrate height map is needed, which we extract from real papers through a shape-from-shading technique (Section 5.3). With this acquired data, the artist can first control the substrate roughness through a global scaling factor r that modulates the depth/height of the original height map $H \in [0, 1]$ (Equation 5.2).

$$H_r = ((H - 0.5) \times r) + 0.5\,. \tag{5.2}$$

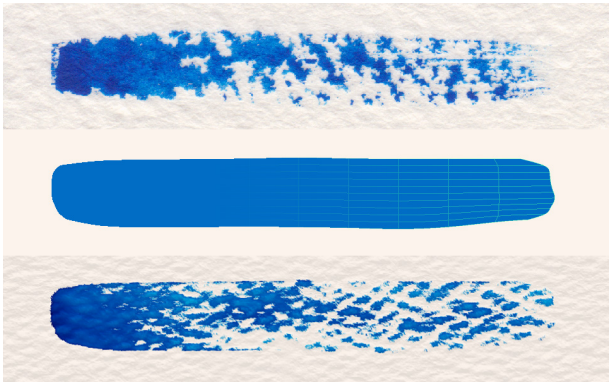


Figure 5.4: Results of dry-brush and granulation in object-space. From top to bottom: scanned real watercolor, 3D render, watercolorized 3D render.

With the overall roughness of the substrate height map, we calculate the pigment application. Since the dry-brush effect is closely related to the substrate granulation and they cannot happen at the same time, these two effects are consolidated within a joint pigment application procedure. Granulation tends to show throughout watercolor paintings, but it does so at different degrees, benefiting from an art-directed local placement. On the other hand, dry-brush is strictly art-directed and needs to be placed consciously in certain parts of a painting. The application of both effects is locally controlled through a parameter a_{fx}, which can be applied procedurally and/or art-directed by the artist by painting on the 3D geometry. The resulting pigment application P_a is governed by Equation 5.3.

$$P_a = \begin{cases} t_{dry}|H_r - a_{fx}|(C_s - C) + C & \text{if } H_r < a_{fx} \\ C^{d_a} & \text{if } H_r \geq a_{fx} \end{cases} \tag{5.3}$$

If the application parameter a_{fx} is greater than the elevation of the height map H_r (case 1 in Equation 5.3), a dry brush effect is applied through a linear interpolation between the substrate color C_s and the original color C, with a global dry-brush threshold t_{dry} to smoothen the dry edge transition. Otherwise (case 2 in Equation 5.3), substrate granulation happens, darkening the color C by the accumulated density d_a. Since granulation tends to show throughout watercolor paintings, the application density d_a is calculated as $d_a = |a_{fx}| + d_{\min}$—with $d_{\min} = 0.2$ representing the default granulation over the image. Then, to empirically augment the darker concentration of pigments at brighter colors—C^{d_a} will not darken them much—, we increase the granulation density ($\times 4$) through a linear interpolation of the density contribution with the color luminance L (Equation 5.4).

$$\begin{aligned} d_a &= L(d_a \times 4 - d_a) + d_a \\ &= d_a(3L + 1). \end{aligned} \tag{5.4}$$

The final pigment granulation density d_a is governed by Equation 5.5, where D is a global density parameter affecting the entire image.

$$d_a = 1 + (d_a \times D \times (1 - H_r)) \,. \tag{5.5}$$

The accumulated density d_a is eventually processed by Equation 5.3 to obtain the granulated pigment application P_a. Dry-brush and granulation can thereby be art-directed to the user's content, applied into simple 'dry' gradients (Figure 5.4) or even used as an indication of leaves (Figure 5.5d). When used in conjunction with different substrates and under the influence of a strong substrate distortion (Section 5.3.1), dry-brush can also form abstracted dry patterns (Figure 5.5e).

(a) Dry-brush tree © Joe Cartwright

(b) Tree mesh

(c) Tree mesh with painterly material

(d) Dry-brush variant 1

(e) Dry-brush variant 2

Figure 5.5: Recreating a traditional watercolor painting using a dry-brush application to indicate leaves.

5.2 Edge-based Effects

In any type of imagery, edges are imperative as they delineate shape, define form and create contrast. Because of these properties, edges are used in many computer science applications (e.g., image processing, computer vision and image synthesis). Edge-based effects are common in most natural painting media and are widely used for stylization purposes. However, in watercolor imagery, edges present characteristic phenomena due to the fluidity of the medium. We address two edge-based effects commonly found in watercolors: *edge darkening* and *gaps & overlaps*.

When watercolors dry, pigments are often carried out to the outer rim of the painted wet area, concentrating pigments and creating a darkened edge (Section 5.2.1). Edge darkening serves as a reminder of familiar organic qualities of traditional materials and add tonal contrast for the definition of forms. These effects are commonly observable in traditional watercolors and portrayed in the literature [Brown, 2007; Soan, 2014]. Applying a combination of edge-based techniques to excellent artistic effects is demonstrated in the 2014 painting by Frits Ahlefeldt (Figure 5.6).

Depending on the style of the painting, when painting adjacent areas, neighboring colors tend to present gaps or overlaps between each other (Section 5.2.2). The sophistication of the color palette is increased by combining colors at different levels of transparency at the edges through overlaps. Conversely, introducing elements of striking luminance by revealing the substrate at edges through gaps, adds further options for artistic refinements.

Figure 5.6: Sea Turtle in Watercolor, © Frits Ahlefeldt, 2014.

These effects present themselves as special challenges, because of their reliance on the relevant edges that a watercolorist would stylize. For this purpose, edges first need to be detected in the image. Many algorithms are available [Papari and Petkov, 2011], but the most common and computationally efficient ones involve the use of filters to compute local differentials such as the *Sobel*, *Canny* or *Difference of Gaussians* (DoG) filters. However, applying them to regular RGB images only detects discontinuities in the color gradients. They miss edges in 3D space that

share similar tonalities—but are perceived through our stereo vision. To detect them, we run a *Sobel* filter on an RGBD (Red, Green, Blue and Linear Depth) buffer to generate meaningful edges [Nienhaus, 2003; Saito and Takahashi, 1990]. We further control the contribution of the depth channel by a parameter, increasing its influence on the edge detection and generating more uniform edge intensities, regardless of their color tonalities. This also enables more coherent edge-based effects, even under motion when neighboring colors change. Once the edges E have been extracted (Figure 5.7), they are used to synthesize edge darkening and gaps & overlaps.

RGB Depth RGBD edge E

Figure 5.7: Breakdown of the edge extraction.

5.2.1 Edge Darkening

Edge darkening is a characteristic effect that manifests itself when the pigment is carried out to the boundaries of a colored area, due to surface tension. This causes a gradient concentration of pigments towards the edges of painted areas and, therefore, a darker appearance. The edge darkening effect has been one of the most studied and implemented characteristics in watercolor.

Lei and Chang [2004] proposed to darken edges in image-space using a *Sobel* filter and subtracting the detected edges from the image color. Their approach darkens the edge, but it does not concentrate the color or showcase a gradient accumulation of pigments.

Burgess et al. [2005] improved upon this by proposing the use of color and depth to find edges. A set of 25 sparse neighboring pixels were sampled and, depending on their differences and a set of thresholds, an edge was identified. The color was then modified using the *Kubelka-Munk* color model. Their approach concentrates the color at the edges, but edges do not present a gradual pigment accumulation.

Bousseau et al. [2006] resorted to the use of a simple symmetric kernel to detect edges, which were later used to concentrate the pigmentation using their proposed color transmittance modification model. Similarly to Burgess et al. [2005], their

approach concentrated color at the edges, but does not incorporate the gradient pigment accumulation that happens with the natural medium.

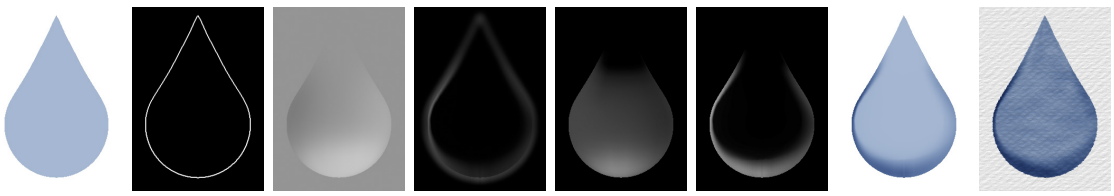
Luft and Deussen [2006a] used a *Difference of Gaussians* to subtract the alpha contribution of segmented elements, leaving the edges with the original color, but lightening the inside of the segments, instead. While this approach does feature a gradient pigment accumulation, it does not darken the actual color at the edges.

Wang et al. [2014] combined the approaches of Lei and Chang [2004] and Bousseau et al. [2006] to use a *Sobel* edge detection to concentrate the color using the color transmittance modification model. While providing a better edge detection and accumulating the pigment colors, no gradient accumulation was synthesized.

Edge darkening has been widely studied in previous work, yet, we take advantage of an RGBD edge detection and extend the detected edges to generate gradient pigment accumulation using the color transmittance modification model—while offering individual local control over the darkened edge width and intensity.

Although the *Sobel* filter produces sharp edges (Figure 5.8b), we generate gradient edge darkening by subsequently blurring them. The convolution is controlled and performed through a separable *Gaussian* blur kernel, which is established at each individual pixel by an edge width parameter W_k, painted on the surface of the object (Figure 5.8c). The resulting blurred edges ($E_b = G_{W_k} * E_{sobel}$, Figure 5.8d) are then amplified by a global edge darkening value and painted edge intensity parameters I_e (Figure 5.8e). This operation results in the final edge density E_d (Figure 5.8f—actual density reduced for illustration purposes) to darken the color through the color modification model $C_{dark} = C^{1+E_d}$, where C is the original color (Figure 5.8a). The edge darkening contribution on the final watercolorized result can be seen in Figure 5.8h.

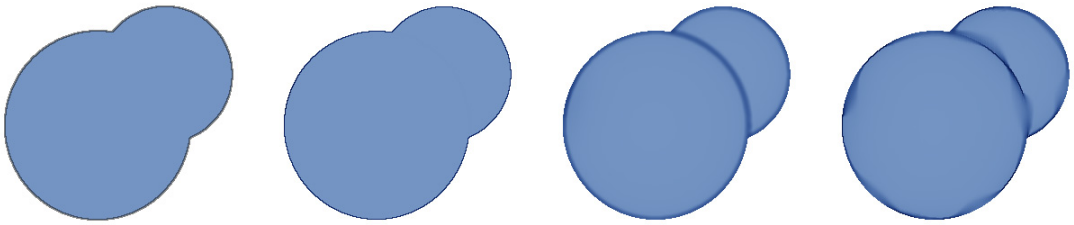
The resulting edge darkening effect is fully controllable and can be coherently art-directed by an artist for any view point, taking advantage of the object-space painted parameters seen in Figures 5.8c and 5.8e. The addition of procedural noise to the parameters can further add fluctuations to the edge darkening width and intensity throughout the object, providing more natural variations.



(a) C **(b)** E_{sobel} **(c)** W_k **(d)** E_b **(e)** I_e **(f)** E_d **(g)** C_{dark} **(h)** Result

Figure 5.8: Breakdown of our art-directed edge darkening.

The versatility of our edge darkening approach can be seen and compared in Figure 5.9. In this comparison, previous approaches only provide a uniform width and struggle to darken edges with similar colors. We significantly improve upon these approaches and at the same time offer extensive control over our synthesis.



Lei and Chang [2004] | Wang et al. [2014] | Our approach | With art-direction

Figure 5.9: Different approaches towards edge darkening.

5.2.2 Gaps & Overlaps

Gaps are characteristic effects of watercolor that materialize when the artist does not want the colors of neighboring regions touching each other, either because they are both wet and might bleed into one another, or for aesthetic preferences. Conversely, overlaps are characteristic effects that happen when the artist does not mind these colored areas from slightly overlapping each other, mostly for aesthetic reasons or accidental deviations when painting. Small gaps & overlaps may also appear as a natural byproduct of hand-tremors, which are involuntary reflexes in the human nervous system. These effects are often present in watercolor illustrations, but they have rarely been studied in image- or object-space approaches.

Luft and Deussen [2006a] explored the effect by rendering each group of elements in individual layers, distorting these independently and compositing them together

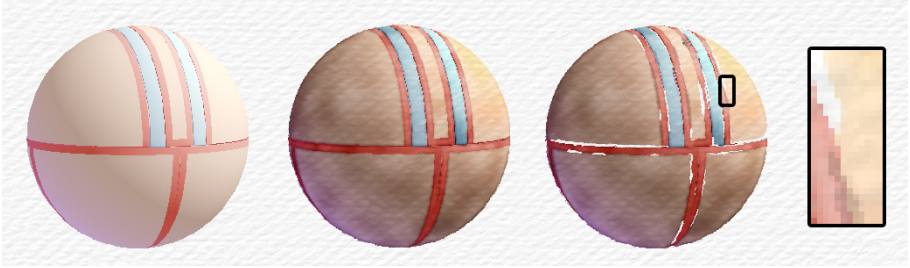


Figure 5.10: We can synthesize gaps & overlaps to achieve a sketchier watercolor look. From left to right: original colors, our results without and with gaps & overlaps.

afterwards. In this way, the distorted layers may partially overlap or show gaps between them. Nonetheless, the approach is quite limited, as objects within a common layer will not present any gaps or overlaps. Additionally, relying solely on image-space computations to synthesize this effect introduces spatial incoherences under animation.

Wang et al. [2014] proposed to synthesize gaps & overlaps from hand tremors, where edges are distorted due to involuntary fluctuations of the human nervous system. This is accomplished by distorting boundaries through low frequency *Perlin noises*. While attaining good randomized results, spatial coherence issues are embedded in this approach.

To synthesize gaps & overlaps, we propose a novel approach to distort the rendered image at the vicinity of its edges by either making the substrate appear between two adjacent regions, or picking and blending neighboring colors. Our method addresses the limitations of other approaches and is guided by art-directed parameters, painted by the user on the surface of the stylized meshes and/or through procedural parameters mapped onto the object.

As gaps & overlaps are located nearing edges, these effects highly depend on a robust edge detection. The previously detected edges have a general width of two pixels—one pixel on both sides of the color or depth discontinuity (Figure 5.11b). Since gaps & overlaps may be wider than this (Figure 5.11c), we extend the detected edges to a maximum constant width m using a linear filtering kernel (Figure 5.11d). Marching along the gradient of those linear edges, we converge to adjacent colors (Figure 5.11e) for either overlapping colors (Figure 5.11f), or generating gaps (Figure 5.11g). Depending on the linear edge values, some gradients might not converge towards neighboring contrasting pixels. Fortunately, these cases are mostly found at immediate edge boundaries, which can be identified with the original unwidened edge boundaries. The entire algorithm is described with pseudo-code in Algorithm 1 and the results presented in Figures 5.10, 5.12 and 5.13.

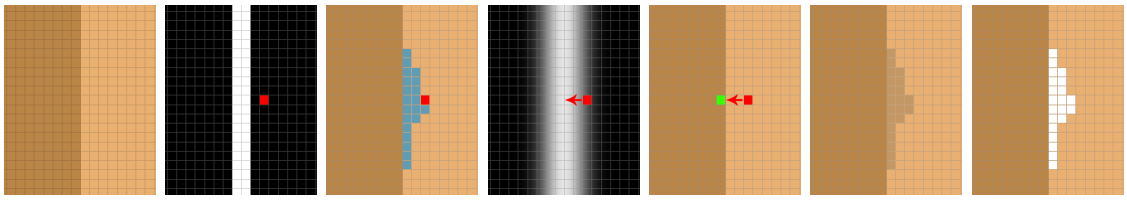
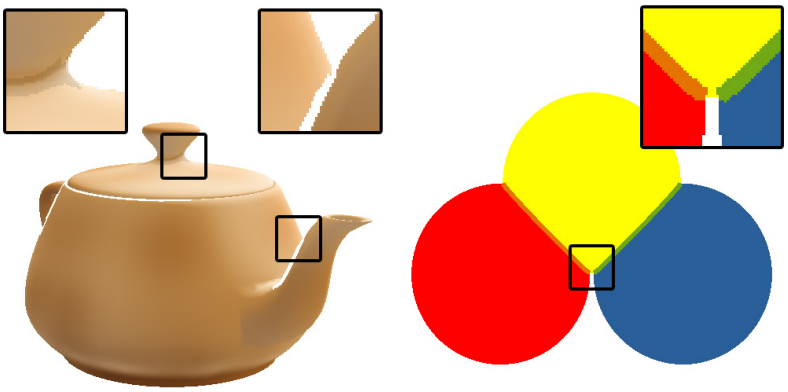


Figure 5.11: Breakdown of our gaps and overlaps approach.

To enhance the overlaps generated by our approach, the mixing of colors is done in RYB (Red, Yellow, Blue) space [Gossett and Chen, 2004], following the brightness-preserving color mixing model described by Chen et al. [2015] (Fig-

ure 5.12b). The gaps can show either the substrate or the color of the revealed 3D models, if the effect is implemented such that the gaps modify the alpha channel of transparent objects, showing any color underneath.



(a) Overlaps - Gaps **(b)** Mixed color overlaps

Figure 5.12: Results of gaps & overlaps in object-space.

Figure 5.13: Stylized object featuring gaps & overlaps to accentuate a sketchier look. Lowpoly House model, CC BY Rafael Scopel.

Algorithm 1: Gaps & overlaps pseudo-code.

Input: RGBD image I, extended edge image E_e, edge image E, substrate color C_s, gaps & overlaps parameter $p \in [-m, m]$
Output: Gaps & overlaps image I_{go}

for all pixel $I(x, y)$ **do**
  $I_{go}(x, y) = I(x, y)$
  // Check if not substrate color
  if $(I(x, y) \neq C_s)$ **then**
    // Check if in extended edge
    if $(E_e(x, y) > 0.2)$ **then**
      $\vec{G} = normalize(\Delta_u, \Delta_v)$ // Get gradient by fetching neighbor pixels
      // **OVERLAPS**
      if $(p > 0)$ **then**
        // Check up to m pixels along $\vec{G}$
        for $i \in [1 : m]$ **do**
          // Check if enough overlap
          if $(p < i)$ **then** break
          $C_n = P(x + i\vec{G}_x, y + i\vec{G}_y)$ // Get neighboring color
          // Check for difference in RGBD space
          if $(\|C_n - I_{go}(x, y))\| > 0.5)$ **then**
            // If not substrate, mix
            if $(C_n \neq C_s)$ **then**
              $I_{go}(x, y) = \text{mixRYB}(I_{go}(x, y), C_n)$ // Mix colors in RYB space
            break
        // Loop reached max → direct edge
        if $(i == m)$ **then**
          // Gradient did not converge to another color
          $C_n = P(x - \vec{G}_x, y - \vec{G}_y)$ // I(x,y) at E edge → negate $\vec{G}$
          $I_{go}(x, y) = \text{mixRYB}(I_{go}(x, y), C_n)$ // Mix colors in RYB space
      // **GAPS**
      if $(p < 0)$ **then**
        // Check if direct edge
        if $(E(x, y) > 0.7)$ **then**
          $I_{go}(x, y) = C_s$
        else
          // Check up to m pixels along $\vec{G}$
          for $i \in [1 : \text{m}]$ **do**
            //Check if enough gap param.
            if $(|p| < i)$ **then** break
            $C_n = P(x + iG_x, y + iG_y)$ // Get neighboring color
            // Check for difference in RGBD space
            if $(\|C_n - I_{go})\| > 0.5)$ **then**
              $I_{go} = C_n$ // Assign substrate color
              break

Note: Transparent gaps are obtained by zeroing the output alpha channel instead of using the substrate color.

5.3 Substrate-based Effects

Be it the canvas on which oil paint is placed, or the paper where watercolor pigments are settling in, the substrate has a significant role in natural painted media. The substrate can shine through, distort, accumulate pigments or shade the surface in ways that alter the actual painting. Watercolors, by being a translucent fluid medium, is especially sensitive to the substrate it is being painted on, and many effects are significantly affected by it.

Ideally, an accurately measured paper profile would provide the required information to synthesize these characteristic effects. However, profilometers for accurate surface height measurement, either optical or stylus-based, are not easily accessible—the equipment is expensive and scanning may take a significant amounts of time. For lack of a profilometer, we resorted to scanning three watercolor papers with a flatbed scanner along multiple lighting directions, and acquired their profile through a shape from shading technique [Barmpoutis et al., 2010]. While the results might not be as accurate as those from an actual profilometer, most irregularities within the extracted 3D profile geometry can be fixed in object-space (i.e., uneven height along the scanned surface).

Once the 3D substrate data is extracted, a height map H and normal map N can be easily produced (Figure 5.14). This data is embedded into a single texture file, where the surface inclinations from the normal map are stored in the red and green channels (U and V i.e., horizontal and vertical), and the height map is embedded in the blue channel. We use these substrates to help synthesize pigment-based effects such as the dry-brush application and pigment granulation (Section 5.1.2), and to synthesize substrate-based effects such as the substrate distortion and add the possibility to generate interactive substrate lighting over the synthesis.

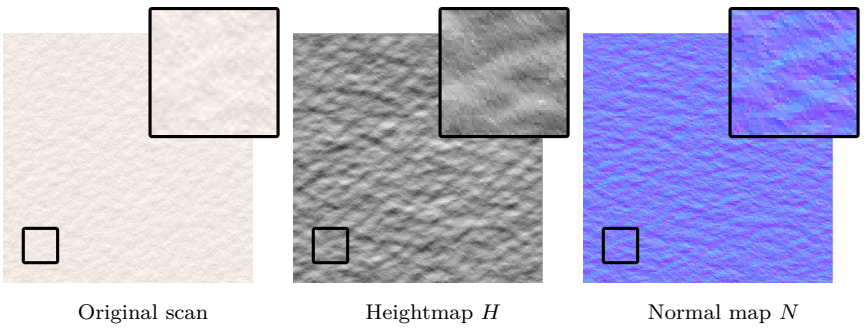


Figure 5.14: Scanned substrate and extracted profile data.

5.3.1 Substrate Distortion

Substrate distortion, often referred to as *wobbling effect*, occurs when the rough profile of the substrate shapes small distortions to the areas being painted. This happens through the interaction between the roughness of the surface and the hairs of the brushes, or through surface inclinations that move the pigments towards the valleys of the paper.

Curtis et al. [1997] recognized the importance of substrates early on and took advantage of a procedural set of papers, generated through *Perlin noise* [Perlin, 1985] and *Worley noise* [Worley, 1996], to drive their physically-based approximation. Substrate distortion was successfully achieved as a result of their fluid simulation, but the substrate was unfortunately not physically accurate. Therefore, the remaining approaches all took advantage of scanned papers.

Lei and Chang [2004] distorted the image according to the luminance deviation of the paper at a given pixel, compared to the mean luminance of the paper. Bousseau et al. [2006] took the horizontal and vertical gradients of the paper texture to distort the image. Luft and Deussen [2006a] directly modulated the intensity of the image at the borders, getting rid of elements at a certain threshold. While each of these empirical approaches addresses substrate distortion differently, the approach of Bousseau et al. [2006] comes closest to the actual physical behavior. However, they are all based in the intensity of a scanned paper texture, which does not contain the surface intricacies of an actual paper profile.

We improve upon this, by relying on our extracted paper profile data to directly distort the resulting image using the subtrate's UV inclinations. We can also define the amount of distortion locally and/or procedurally, which can generate additional uses, but brings depth into consideration.

In our approach, paper distortion is effectively computed by directly sampling the color values at UVs that have been shifted by the surface inclinations, available through the normal map N of the paper texture. The amount of paper distortion can be art-directed towards any desired result. This can be especially useful when using substrate distortion to depict certain effects, such as the intricacies and reflection/refraction of the water surface (Figure 5.15). However, a heavily distorted area should not distort towards a less distorted area that is in front (in 3D space). To allow such usage, we add a further depth-test to our UV sampling that will first check if the destination is in front of the currently distorted pixel by comparing its values in the depth buffer. If the destination is in front, the distortion amount of the destination will take place, instead.

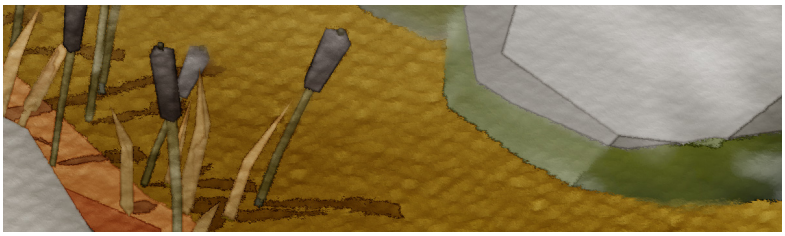


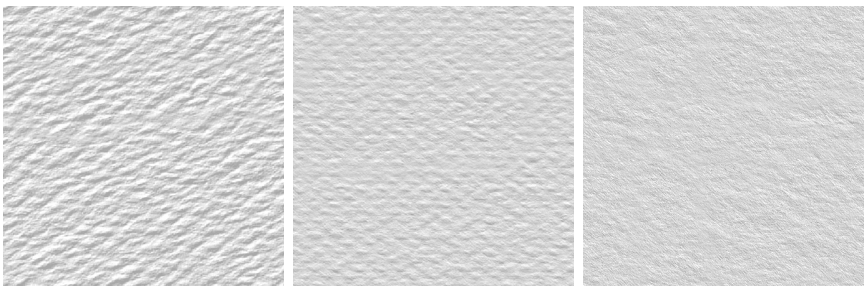
Figure 5.15: Close-up of Figure 5.23f rendered at 8K. Substrate distortion is used to depict water, but does not distort towards elements that are in front.

5.3.2 Substrate Lighting

A painting is always affected by external lighting conditions, thus it is of importance to consider the substrate lighting that would affect the final watercolorized imagery. Unlike oil paintings [Hertzmann, 2002], in the case of watercolors, the profile of the paper tends to remain mostly unchanged. The carrier evaporates and the remaining pigmentation has generally little effect over the substrate profile—meaning that a shaded watercolor painting will mostly be affected by the roughness of the substrate. Since the normals of the physical substrate were previously acquired, deferred shading can be easily conducted on top of the painted image, emulating external lighting conditions on the watercolor painting and increasing the tangibility of the digital substrate (Figure 5.16).

Due to unavailable substrate profile data, this effect could previously not be controlled and was fixed to pre-existing shading of the scanned substrate textures. With the extracted substrate profile and normals, we implement a simple diffuse illumination model I_d that can easily be customized through a lighting direction $\vec{L}$, the extracted normal map $\vec{N}$, the substrate roughness r and the diffuse shading contribution d_s through Equation (5.6).

$$I_d = 1 - (d_s(1 - (\vec{L} \cdot (r\vec{N})))) \,. \tag{5.6}$$



Arches-Rough
light dir.: top-left

Fabriano-Cold Press
light dir.: top

Burgund-Matt
light dir.: top-right

Figure 5.16: Three watercolor paper specimens lit from different angles.

5.4 Abstraction-based Effects

Abstraction, or lack thereof, is arguably one of the main aspects that define an artist's style. As such, abstraction is a broad topic that can happen at multiple stages within 3D content (e.g., modeling, texture/color, lighting, rendering, compositing). We specifically address the synthesis of abstraction-based effects, characteristic of the traditional watercolor medium. Watercolor is a fluid medium that is prone to disperse and combine with nearby colors, creating an effect commonly known as color bleeding, which can abstract colors and shapes together (Section 5.4.1). This effect cannot be reproduced traditionally by any other medium, except for maybe diluted acrylics [Smith, 2003].

A wet-in-wet application happens when pigments touch a wet area and disperse through the wet surface. While a wet-in-wet application could be considered the simplest form of color bleeding, the bleeding effect is rather associated with colors dispersing into each other. As the colors of objects are already set in object-space by the painterly material (Chapter 4), color bleeding can be applied at a later stage, combining colors within or even between rendered objects.

To synthesize color bleeding empirically, we need to find a controllable way to gradually blend nearby colors into each other within a specific area. However, colors should not bleed from certain art-directed areas and other effects should not materialize when colors are bled (i.e., edge darkening).

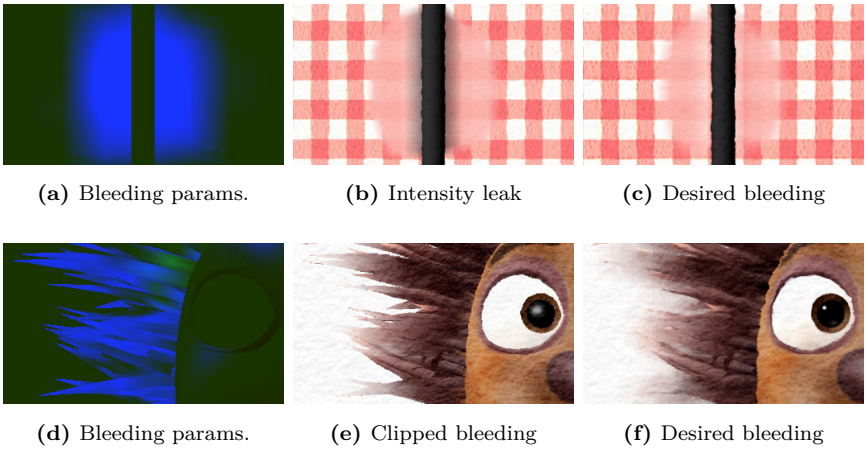
5.4.1 Color Bleeding

Color bleeding is mostly associated and evident between two wet areas that bridge the surface tension, allowing pigments from either side to flow into each other. This behavior gradually combines colors from one area to the other, creating smooth transitions between the previously existing edge. As such, color bleeding is a localized effect that is, for the most part, consciously placed and driven by the artist. As most approaches have not taken art-direction of effects into account, this characteristic effect has only been empirically addressed by one researcher to date.

Wang et al. [2014] proposed an image-space approach that randomly scatters pixel-sized seeds along boundaries and filters these areas with an ellipse-shaped kernel oriented along the normal vectors of the boundary edge. These seeds contain the darker color within the boundary and are only scattered along the brighter side

of it. The ellipse-shaped kernel blends the colors together, achieving feather-like color bleeding that only forms if a certain set of rules is met. While the bleeding results of the approach are satisfactory for the casual user, an artist will want to locally art-direct the effect, which brings special considerations. Additionally, randomly scattering seeds will not be spatially or temporally coherent.

We approach color bleeding from a localized, art-directable perspective. The simplest way to blend colors together is by using a big blurring kernel that will sample and mix all neighboring colors in bled regions (e.g., blue channel in Figures 5.17a and 5.17d). While straightforward to implement, a naïve blurring approach has two fundamental problem cases: (1) when objects in the background are bleeding, but are obstructed by a non-bled object in the foreground and (2) when objects in the foreground are bleeding along their silhouettes over a non-bled background. These problem cases are illustrated in Figure 5.17



(a) Bleeding params. **(b)** Intensity leak **(c)** Desired bleeding

(d) Bleeding params. **(e)** Clipped bleeding **(f)** Desired bleeding

Figure 5.17: Images (a) and (d) are the painted bleeding parameters; (b) and (e) are the results from a naïve bleeding approach; (c) and (f) are results using our bleeding algorithm.

The first problem case presents itself as an intensity leak, which is caused when bled pixels are influenced by all neighboring colors (Figure 5.17b). In this example, the black line at the *foreground* has not been given any bleeding parameters (Figure 5.17a) and should, therefore, not be considered when something at the *background* is bleeding (the bleeding is occluded). However, a naïve blurring approach considers all pixels equally and does not respect different bleeding parameters or depths. The second problem presents itself as clipped bleeding at the silhouettes, which is caused when pixels without bleeding parameters do not consider nearby pixels that might be bled (Figure 5.17e). In this example, the spikes are bled according to the bleeding parameters (Figure 5.17d), but are constrained by the parameter edge (i.e., the bleeding does not flow from the spikes into the background color). This clipping would only be desired, if the bleeding mask was obstructed by a *non-bled* object at the *foreground*, as in the previously elaborated case. However, when the object at the *foreground* is the one bleeding, it should spread the color

into objects found at the *background*. A naïve blurring approach does not consider pixels outside of the bleeding parameters. Figures 5.17c and 5.17f show the desired outcome of the elaborated problems. These results have been rendered by our color bleeding approach, which is featured next.

A 4D joint-bilateral filtering algorithm was derived to include and solve these problem cases, considering the depth and bleeding control values, in addition to the two spatial dimensions x and y. The bilateral filtering algorithm, coined by Tomasi and Manduchi [1998], is therefore adapted to blur the image, while respecting depth and bleeding control differences/edges to solve the previously stated problem cases. To this purpose, two rules are followed:

- Bled pixels should not leak from un-bled foreground pixels
- Bled pixels should bleed into/from un-bled background pixels

This reasoning approaches a layered application of traditional watercolors, in which a wet-in-wet application of a foreground object will bleed into the background, whereas a wet-on-dry application of a foreground object should not bleed into a previously bled layer underneath. A pseudocode of the proposed algorithm is given in Algorithm 2.

Since the blurring in our approach is performed with a fixed kernel, the resulting bled image B is finally integrated into the color image C by blending it through to the bleeding mask b_{fx}—which was modified by the algorithm to include the surrounding bled pixels. This final step synthesizes the pigment diffusion, which occurs when the colors bleed into each other. The effect and the bleeding parameters (blue control channel) can be clearly observed in Figure 4.10. Further examples of color bleeding are integrated in Figures 5.1, 5.22 and 5.23.

Algorithm 2: Joint-bilateral color bleeding (two-pass)

Input: color image **I**, depth image **Z**, control image mask **C**, one-dimensional Gaussian normalized kernel weights **W** ($size = 21$, $\sigma = 20$) and depth threshold **T**
Output: Bled image **B**, control image mask **C**

```
    for all pixel I(x, y) do
2:      for i ∈ [-10 : 10] do
            // Check if source or destination pixels are bled
4:          if ((C(x, y) > 0) or (C(x + i, y) > 0) then
                // Initialize bleed and depth queries
6:              bleed = False
                sourceBehind = False
8:              // Check if source pixel is behind
                if (Z(x, y) − T) > Z(x + i, y) then
10:                 sourceBehind = True
                // Check bleeding cases
12:             if C(x + i, y) > 0 and sourceBehind then
                    bleed = True
14:             else if C(x, y) > 0 and not sourceBehind then
                    bleed = True
16:             // Bleed colors
                if bleed then
18:                 // Add weighted destination color
                    B(x, y) = B(x, y) + I(x + i, y) × W[i + 10]
20:             else
                    // Add weighted source color
22:                 B(x, y) = B(x, y) + I(x, y) × W[i + 10]
            else
24:             // Add weighted source color
                B(x, y) = B(x, y) + I(x, y) × W[i + 10]
26:     //Expand control mask
        Modify C(x, y) as bled, if bleeding took place
```

Compute another pass with a vertical kernel using the modified control image mask, which includes the influences of previous horizontally bled pixels.

5.5 Discussion

The watercolor stylization brought a novel art-directed approach towards the watercolor synthesis in object-space. While considering art-direction raised additional challenges in the synthesis of some effects (i.e., color bleeding, substrate distortion), these could be solved by involving additional object-space data. Our approaches manage to synthesize the essential characteristic effects (Section 2.1.2) with new techniques and improved methods, however, a successful watercolor portrayal requires individual stylization effects to be directed together in a congruent manner.

For this purpose, I stylized various objects and scenes during my studies, some of which can be seen throughout Figure 5.23. Additional rendered imagery, from professional CG artists, which participated in our first user study, can be found in Figure 5.22. Finally, a comparison to other object-space watercolor simulation systems can be observed in Figure 5.20.

The watercolor characteristics that we have contributed in this chapter extend the palette of watercolor effects found in object-space by introducing novel approaches towards color bleeding, dry-brush and gaps & overlaps. Together with this contribution, we enhanced and improved upon existing approaches by synthesizing a robust edge darkening with gradual pigment accumulation, efficient and depth-aware substrate distortion, accurate substrate granulation and controllable substrate lighting. All these effects are complemented by real-time art-directed localized and procedural control in object-space. Integrating these watercolor effects at different levels of control and with immediate feedback is addressed by our direct stylization pipeline (Chapter 6).

At this point, it is important to emphasize that, in the same way as traditional watercolors are used by acclaimed artists such as Anders Zorn and Steven Kozar to depict realism, a watercolor stylization can also be used to simulate watercolorized photorealistic depictions. This happens when using highly detailed geometry and photorealistic textures, as the stylization does not pre-filter them. Therefore, the level of abstraction and the degree of photorealism in the stylized result depends mostly on the form of the objects and the textures assigned by the user. This implementation supports a wider spectrum of watercolor imagery, but relies on the artist to source the amount of realism that is rendered. A simple, but clear example that illustrates how the watercolor stylization processes photorealism can be found in Figure 5.18. In this comparison, both spheres are rendered with the same lighting conditions and albedo/normal/specular texture maps. However, the aesthetic results are quite different. The painterly watercolor reflectance model and its attributes, together with the art-directed effects, give the stylized example in Figure 5.18b a watercolorized look. When observed in detail, the shade tint

provides additional color variations, the edge darkening accumulates pigment, the paper granulation and distortion give a tangible property, whereas the color bleeding gives the result a manual touch. The rendered result, while realistic looking, possesses the qualities of the watercolor medium.

(a) Specular material **(b)** Watercolor stylization

Figure 5.18: Watercolorization example of a detailed surface texture.

5.5.1 Evaluation

Evaluating an expressive rendering approach is no straightforward task [Hall and Lehmann, 2013] and depends on the purpose it is being used for [Isenberg, 2013]. The evaluation of the watercolor stylization is especially challenging, as it ideally involves multiple levels of control in object-space, with the purpose of obtaining aesthetically faithful and pleasing watercolor imagery. Therefore, a specialized skill-set is required from users to fully take advantage of the stylization's potential. These skills include object-space (3D) proficiency, experience within the software and knowledge in the expressive visual style that is being sought after.

The nature of the stylization and the work presented in this thesis is better suited for a qualitative evaluation. Through a qualitative evaluation, we aimed to gain a better understanding of the reception with its intended target users and the effectiveness of the combined effects through our art-directed approach to synthesize watercolor imagery. To this end, two methods were used, which involve: (1) two separate user studies of CG artists using the system by stylizing their own assets and (2) a comparison to previous object-space watercolor stylization approaches using similar assets. From these evaluations, we were able to better understand the limitations of our watercolor stylization and art-directed framework, and iterate based on feedback to improve upon our approach and synthesis. Unfortunately, not all the raised problems could be addressed within my candidature, but these present themselves as a potential ground for future research (Section 7.2).

User studies

Our art-directed watercolor stylization approach is conceived to grant low-level stylization control to the end-user. As such, following the interaction spectrum of *NPAR* proposed by Tobias Isenberg [2016], highly skilled artists are required to fully take advantage of its features and potential. This means that the quality of the aesthetic results does not only dependent on the algorithms and implementation, but is also highly influenced by the skills of the artist using it. An ideal candidate should possess proficiency in:

- Object-space (3D) creation/manipulation in the software of implementation;
- Rendering, composition and watercolor aesthetics;

The skill-set is quite unique and involves both, technical and artistic knowledge. To this purpose, we approached professional CG artists in the animation industry in two separate occasions, to participate in our studies. It can also be argued that traditional watercolor artists would be better candidates for our studies. However, the complexity of working in object-space without previous 3D training and the possibility of testing the system with more varied assets, workflows and animation, supported our decision and resolve.

For our first study, which focused on the watercolor stylization and low-level control (no procedural control had been implemented at this point), a total of seven CG artists from all around the world showed keen interest in using and providing feedback on the watercolor stylization system. Eventually, five of them managed to participate in the user study, investing in total over 51 hours stylizing their own assets. The conducted study first required the users to watch 15 minutes of instructional videos, explaining the different controls available for the watercolor synthesis and the simulated watercolor effects. These tutorials were necessary to bring participants to a similar level of proficiency using the watercolor stylization toolset. Following this, the participants were asked to spend over two hours stylizing their own assets. However, this time-frame did not stop their dedication, with some participants investing more than 15 hours and providing us with their watercolorized imagery (featured in Figure 5.22) and animated clips. To gather the data for this user study, the participants were asked to fill in a questionnaire comprising of 24 questions, designed to study and understand the following:

- The reception of the watercolor stylization;
- The usefulness of a direct stylization pipeline;
- Their needs as artists for a successful watercolor stylization and a direct stylization pipeline.

Regarding the watercolor stylization, only three of the participants had painted with watercolors before, but all participants agreed that the watercolor stylization in object-space satisfied their creative desires (see A-Q3 in Figure 5.19). Some features/effects that participants would like to have improved are: the ability to locally control and embed the watercolor effects in the cast shadows; incorporate local control over custom widths and intensities of the edge darkening effect; incorporate bleed direction and improve upon the temporal coherence of the color-bleeding effect; improve upon the material to support alpha transparency; an easier access to animate the painted parameters; custom paper colors; and a way to export the different passes for final manual compositing. Additional new features/effects that they would like to have included are: object-space splatters; a more subtle solution to temporal coherence for substrate reliant effects; and subtractive color blending. Some of this feedback has already been addressed and included in later versions (e.g., controllable edge darkening, transparency, subtractive color blending in overlaps, animated parameters, custom paper colors), while others provide interesting areas for future research.

None of the participants had experienced or seen object-space watercolor stylization before, but they all agreed that the control over the watercolor effects was enough for their stylization purposes (see A-Q9 in Figure 5.19). Overall, the feedback over the watercolor stylization was positive. The users especially appreciated the way the color bleeding and edge distortion work, together with the ease of use, the amount of personalization, the simple unified tools, the painting of parameters and the immediate visual feedback provided by the direct stylization pipeline.

While the direct stylization pipeline (Chapter 6) has not formally been introduced yet and was still in early stages, we were interested in how our watercolor stylization benefited from real-time, low-level control over the effects. All of the participants strongly agreed on a *Likert* scale (see B-Q3 in Figure 5.19) that the real-time stylization feedback benefited their creative decisions and aspirations. Additionally, to the most part, they also agreed that a direct stylization framework is an improvement over previous methods for object-space stylization (see B-Q5 in Figure 5.19). Since the results are given in real-time in object-space, the interactive workflow enabled the possibility to stylize and navigate the scene within a painted world. This workflow offered them new stylization opportunities to capture details and shots that would probably have been missed otherwise. There was a sound agreement that a direct object-space stylization pipeline is much faster to work with and more artist friendly, compared to traditional pipelines (offline/decoupled) (see B-Q7 in Figure 5.19). However, an option to render the different passes to an offline/decoupled pipeline for a compositing workflow would offer further flexibility and the possibility for custom final tweaks. The combination of both would be an ideal solution for three of the participants.

Overall, the real-time watercolor stylization had an outstanding reception, to the extent of having the participants agreeing that direct stylization systems are possibly the future of artist-friendly stylization tools (see B-Q8 in Figure 5.19). This highlights the relevance of direct stylization pipelines in object-space and motivated us to continue developing the stylization framework. Through this study, we gathered valuable insight on additional features/aspects that would facilitate the artists' workflow within the direct stylization pipeline: the possibility of exporting/importing the stylization settings; having an option of texture map control (instead of vertex color control) for finer bleeding control; a tool to paint textures with traditional looking brushstrokes directly in the viewport; image-space stroke projection (currently in object-space); together with some suggested improvements on the UI.

Regarding Expressive Rendering (NPR) and the implementation of our stylization framework, all but one participant strongly agreed that the implementation in a widely used digital content creation package eased the learning curve (see B-Q1 in Figure 5.19). Finally, four participants strongly agreed that they would start using expressive rendering stylization tools in their own projects, if these were to be available for them (see C-Q1 in Figure 5.19). The entire questionnaire and all responses gathered from this user study can be found in the Appendix. The second user study evaluated the usefulness of each level of control and can be found later in Section 6.1.5.

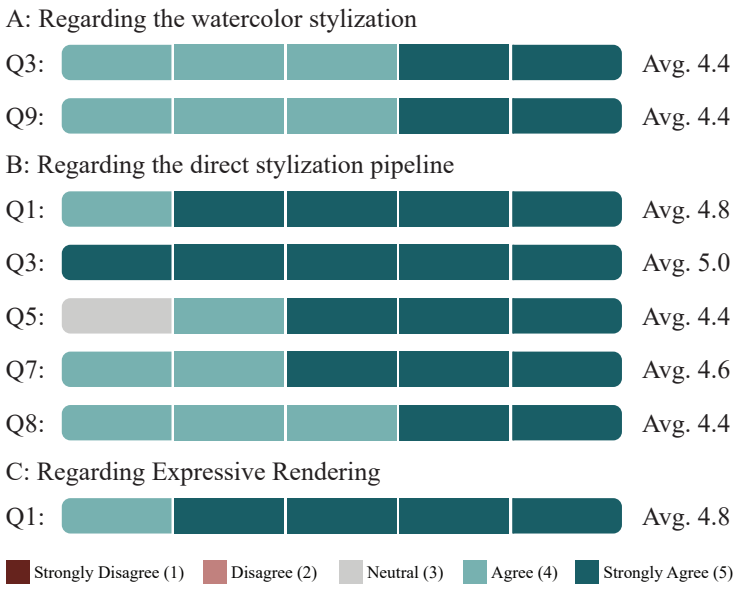
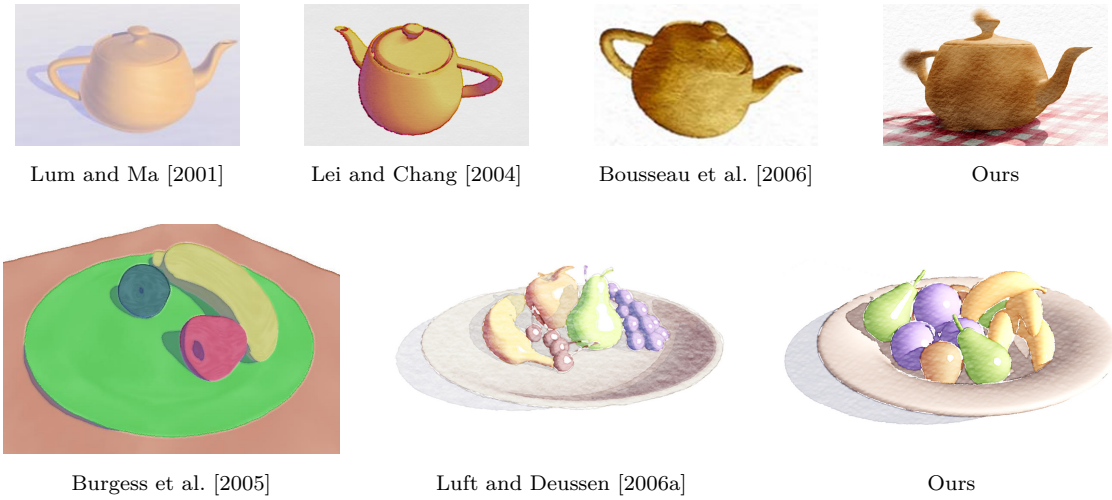


Figure 5.19: Overview of the *Likert* scale responses of the user study.

Comparisons

Most of the characteristic watercolor effects available in object-space, studied by previous researchers, have been addressed in our approach and improved upon. This allows our results to easily assimilate other outcomes and go beyond them, thanks to several unique contributions—all bound and controlled together by the direct stylization pipeline. As it can be appreciated in our results and the comparisons shown in Figure 5.20, the synthesis offers significant flexibility, thanks to its art-direction. The given flexibility results in an overall wider range of characteristic looks found in watercolors.

Previous approaches—addressed in Section 3.3—have mostly focused on recreating a specific style. However, watercolor is a natural medium, which can be used to create artwork in a variety of different ways. The different levels of art-directed control allow to recreate this stylistic freedom, which is complemented through the real-time interaction provided by the direct stylization pipeline. In Figure 5.20, we show previous approaches by other researchers and our results, using similar assets. From these, the watercolorized image at the bottom, to the far right of the comparison, is of special interest, as it was approximated to the results of Luft and Deussen [2006a]. This stylization was conducted as an experiment to test how close an asset could be art-directed, to resemble the look of another approach. While the simulated effects vary and are implemented differently, the results do resemble each other. However, our approach can take full advantage of the art-directed stylization to produce more complex stylized results, as shown in Figure 5.23a.



Lum and Ma [2001] | Lei and Chang [2004] | Bousseau et al. [2006] | Ours

Burgess et al. [2005] | Luft and Deussen [2006a] | Ours

Figure 5.20: Comparisons between watercolor approaches in object-space. Lei and Chang focused on watercolor effects that depict shape. Burgess et al.'s colorful results concentrated on edge effects. Lei and Chang, Bousseau et al., and Luft and Deussen gradually improved the synthesis by incorporating an increasing number of watercolor effects. Our approach consolidated these, proposed new effects and augmented them with art-direction. Please refer to Section 3.3 for a thorough analysis on past approaches.

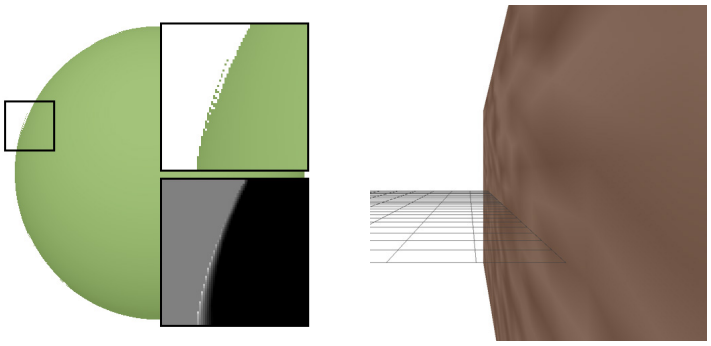
5.5.2 Limitations

Our watercolor stylization is able to synthesize many characteristic watercolor effects, which are essential in watercolor images (Section 2.1.2). However, our approach is still limited compared to the real counterpart. Natural effects such as pigment clumping, hue variations, pigmentation fractals and salt application are still missing in our synthesis and provide exciting avenues for future research. Additionally, traces of brush strokes, which are especially evident in dry-brush or loosely painted subjects, need to be consciously modeled or included within the textures, as our object-space approach will not synthesize these.

The data from the substrate specimens, acquired with a shape-from-shading technique, was accurate enough for our experimental and aesthetic purposes. Yet a proper physical profile scan could present an increased sharpness and fidelity, which could further improve the modeled effects and the overall watercolor look. Additionally, effects that rely on the substrate data for their synthesis (e.g., granulation, dry-brush) are relative to the substrate size. Therefore, effects will look significantly different under diverse substrate profiles and scales, or when the camera moves closer or further away from the subject (if the substrate scale remains static). This behavior is physically correct, but it takes some time to get used to and presents temporal coherence issues under animation. A dynamic paper texture [Bénard et al., 2009; Cunzi et al., 2003; Kaplan and Cohen, 2005] might be able to reduce temporal coherence problems caused by the substrate, with a potential trade-off in fidelity with the original data.

Edge-based effects that rely on the RGBD edge detection are still prone to noise, especially when using photorealistic textures. A general pre-processing stage for textures [Semmo and Döllner, 2015] would create more robust and meaningful edges, especially for effects like gaps & overlaps. The edge-detection would probably also benefit from a deferred rendering pipeline, as the edges would not be influenced by shading conditions (i.e., sharp edges from highlights).

A more general limitation of our method comes from the parameter masks painted on the object surface. While this approach helps the spatial coherence of the render and grants higher control over the effects, the 3D surfaces are subject to geometric (mapping) and perspective distortions (scaling/foreshortening), which may be undesirable. In Figure 5.21a, the spherical mapping of the parameter mask distorts the transition of parameters, increasing its abruptness and potentially creating pixel artifacts. This behavior is especially noticeable when the normals of the surface are almost perpendicular to the view direction (see the bright pixel parameters at the edge of the sphere). In Figure 5.21b, a mask with an otherwise uniform noise pattern is foreshortened through perspective distortion, changing the

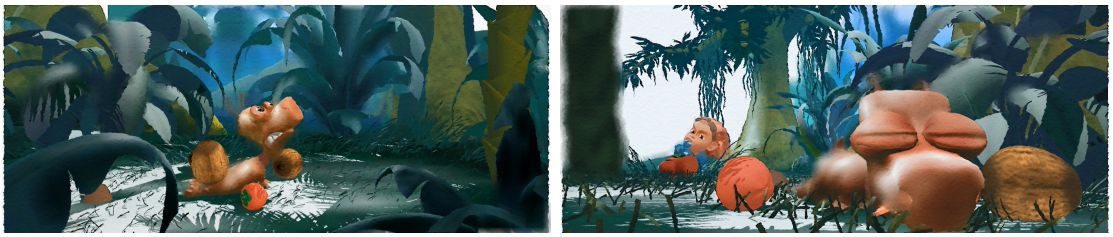


(a) Geometric distortion **(b)** Perspective distortion

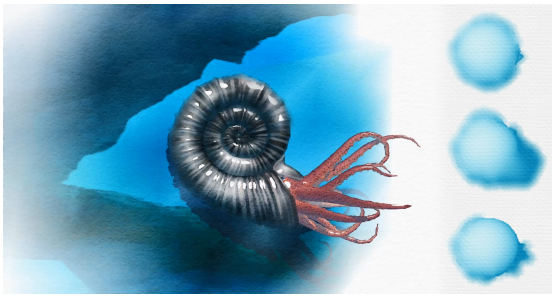
Figure 5.21: Examples of geometric (a) and perspective (b) distortions affecting the parameter masks.

scale of the parameters, relative to their distance from the camera. While these object-space distortions might not be desirable, they are not too noticeable as long as the parameter masks are well designed.

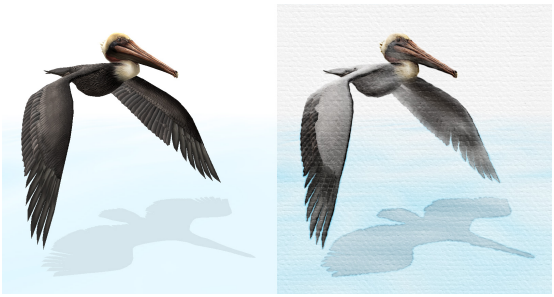
Finally, some implemented effects might present small artifacts, as they have been optimized mainly for performance. However, a combined real-time stylization, together with a high quality offline rendition of the style, should provide the perfect synergy for productions that require polished visuals.



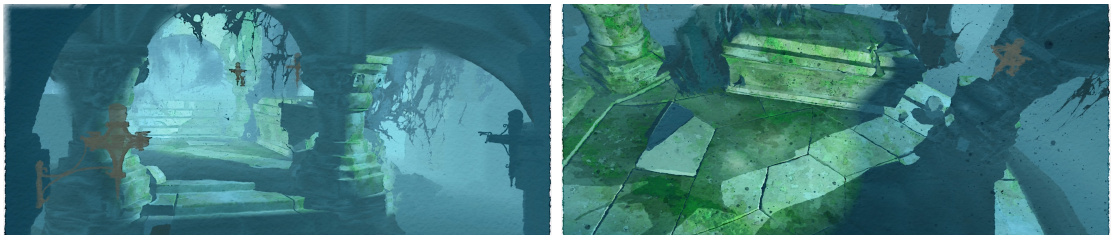
(a) Stylized by Joan and Miquel Cabot



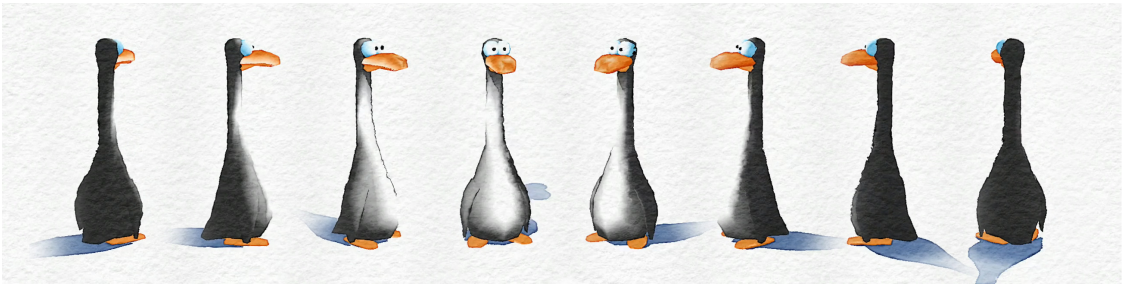
(b) Stylized by Eduardo Altamirano



(c) Stylized pelican comparison (photoreal - stylized)



(d) Stylized by Joan Cabot



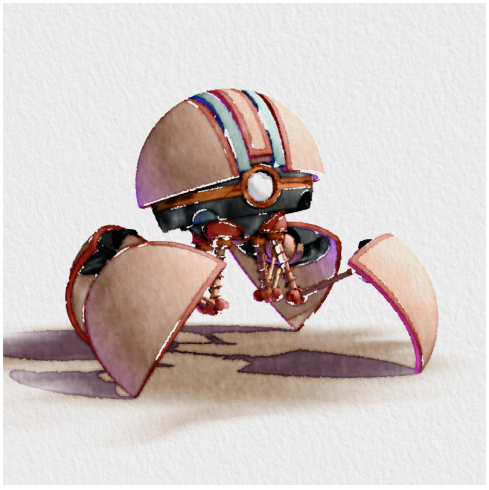
(e) Stylized by Pierre Coffin

Figure 5.22: Real-time renders from different artists using the watercolor stylization. The examples in (a), (b), (c) and (d) were created by participants of our first user study.

(a) Fruit plate model.

(b) Utah teapot model.

(c) Gotita model.



(d) Spherebot model, CC BY Bastien Genbrugge.

(e) Wiz Biz model, CC BY Tom Robinson.

(f) Baba Yaga's hut model, CC BY Inuciian.

Figure 5.23: Stylizing Figure 4.9 with watercolors.

6. Direct Stylization Pipeline

The research presented in this thesis is motivated by the artistic need for an intuitive and generalized approach towards expressive NPR in 3D computer graphics. Therefore, my aspiration was to create a versatile and user-friendly object-space watercolor synthesis, that not only focuses on modeling the characteristic effects of watercolor, but also considers real-time performance and art-direction as a necessary requirement. This chapter focuses on the latter statement. An immediate stylization feedback is crucial as it could drastically reduce the required time for art-direction, facilitating and motivating the creative flow.

Common stylization pipelines for film and television are based on a two-step process (Figure 6.1). First, the required source images are rendered offline from a 3D application, including all necessary stylization intermediates, such as custom masks and additional rasterized object-space data. These images are then gathered and processed in a compositing software to form the final stylized image. In stylized rendering, art-directed changes tend to occur often, but these changes can only be determined once the stylized image is seen and evaluated. If changes are required, some source images usually need to be altered, forcing to return up along the pipeline, back to the 3D application. In the 3D application, the changes are made and the new images are re-rendered and reconfigured back in compositing. This decoupled creative process, while flexible and modular, slows down and even hinders art-direction. Unfortunately, making stylization changes in the 3D application, without immediately viewing the end result, is not intuitive. This makes the process difficult and prone to several repetitions, until the final stylization is achieved. Additionally, the workflow at each stage may involve different artists who might also have different visions and interpretations of the final stylization.

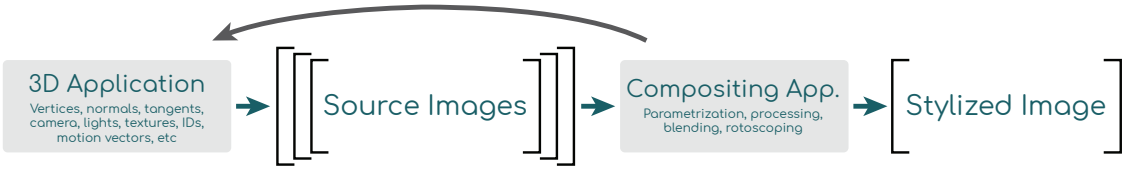


Figure 6.1: Decoupled Stylization Pipeline.

Taking inspiration from real-time technologies and game engine rendering pipelines, a direct object-space stylization pipeline was proposed and developed, which facilitates art-direction through immediate visual feedback (Figure 6.2). This pipeline, attached to a digital content creation software, enables the user to see the stylization results in real-time throughout the creative process—from modeling, to the final art-directed tweaks of the resulting image. This dramatically increases the stylization workflow and helps artists focus directly on the desired result. The stylized image can be taken as is, however, the source images created internally can potentially be rendered offline, taking advantage of ray tracing algorithms. This way, the stylization is art-directed in real-time through the direct stylization pipeline, but the final offline render can be processed through the decoupled stylization pipeline, enabling a compositing workflow with a feature-rich, art-directed and stylized image.

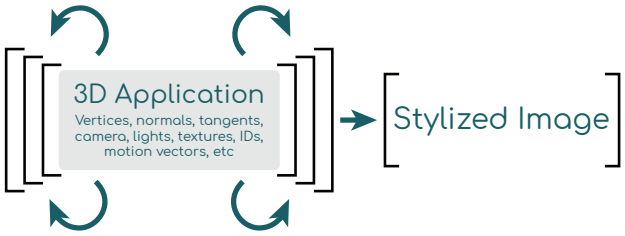


Figure 6.2: Direct Stylization Pipeline.

Art-direction is key towards expressive rendering, however, there is arguably no single solution or tool that will satisfy the expressive requirements of every artist. Each artist has a unique vision, approach and workflow when creating artwork, requiring a varied set of tools. 3D artists have the same requirements, but are often constrained by the available algorithms and digital tools. Developers put much effort in addressing these demands, but the task becomes increasingly complex when considering all the different styles and technical approaches inherent to NPR. During my research, I focused on creating and proposing a generalized framework for filter-based expressive NPR that provides a wide range of tools for artists and that facilitates the further development of art-directed styles for developers. The contributions contained in this chapter comprise the following:

1. an art-directed approach towards covering the interaction spectrum in expressive NPR of 3D computer graphics;
2. a user study with experienced artists and engineers to evaluate the usefulness of each level of control;
3. generalized control semantics to guide cross-stylistic art-direction, which is exemplified for three styles;
4. the implementation insights of the developed direct stylization pipeline: MNPR.

In Section 6.1, I approach the generalized direct stylization framework by covering the interaction spectrum [Isenberg, 2016] for expressive rendering, granting a wide range of control to the artist. By doing so, expressive tools for multiple levels of control are provided, that facilitate the workflow and art-direction of the rendered images. The usability of these tools is then evaluated through another user study.

In Section 6.2, I present stylistic control semantics, which help developers create tools that work predictably with other styles, allowing cross-stylization with the same parameters. Thus, generalizing upon common phenomena and enabling the artist to visualize their expressive results in different styles.

In Section 6.3, I share my thought process and implementation details of this endeavor, which has been open-sourced to enable any future researcher and developer to incorporate their non-photorealistic pipelines in an expressive rendering framework within *Autodesk® Maya®*: MNPR.

6.1 Levels of Control

Research in NPR commonly focuses on algorithms that attain a specific effect or style, but offer limited tools for artists to interact with these effects and modify the look of the stylized imagery. This often restraints the artistic workflow and potential of the algorithm itself. To avoid this, ideally a set of art-directed tools that address different levels of control in real-time are required.

Art-directed tools take time to develop, but the rewards justify the effort. Practical interactive tools can increase the creative involvement of artists who can validate the quality of the algorithm and highlight potential intrinsic problems. An ideal workflow should enable 3D artists to quickly set up a general style using high levels of control, offer tools at the mid-level of control to accelerate the stylization process and also provide the ability to meticulously tweak localized parameters using a low level of control. This would cover the interaction spectrum [Isenberg, 2016] and help the users to achieve their particular vision within the style. The tools to cover this range of control are addressed in this section, as outlined in Figure 6.3.

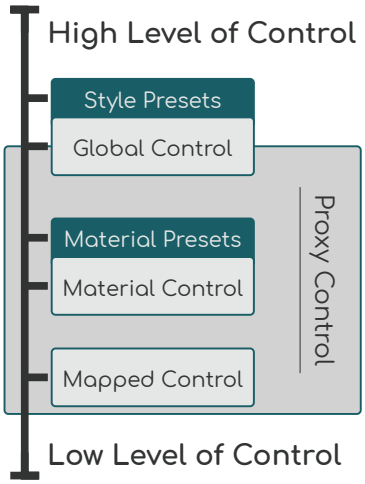
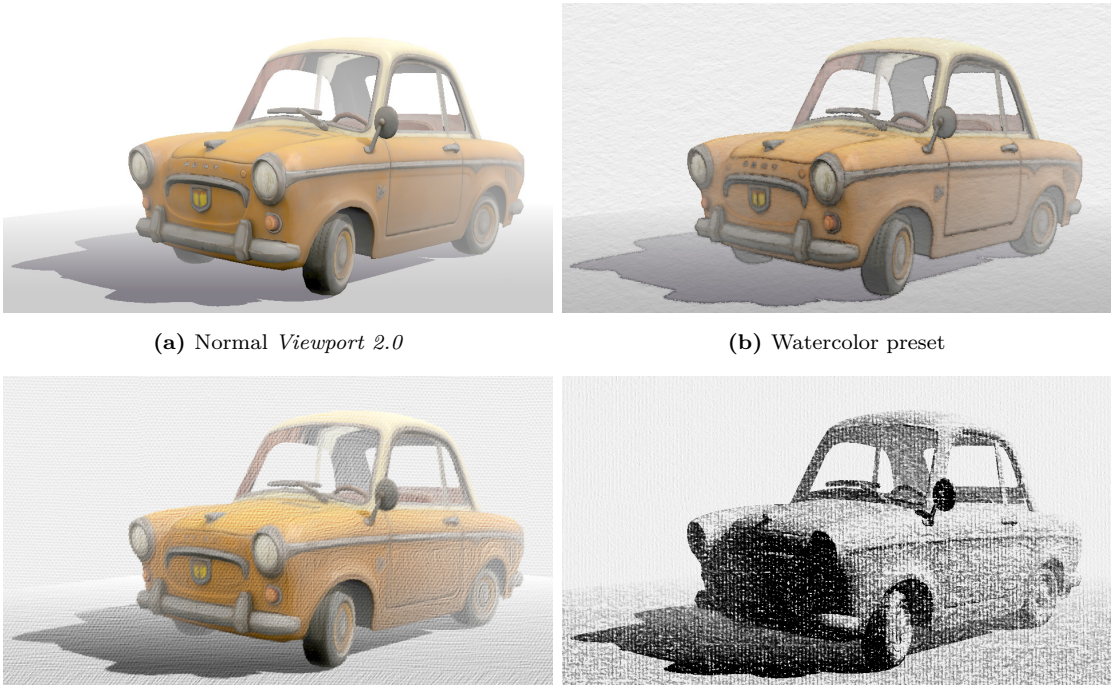


Figure 6.3: Schematic overview of the different levels of control.

6.1.1 Style Presets and Global Control

At the highest level of control, style presets allow the artist to save or load predefined global control parameters—interactively changing between different stylization pipelines (e.g., oil, watercolor, charcoal) and assigning the global control parameters in the entire scene, as illustrated in Figure 6.4. Style presets provide

a fast, but rough start towards a specific style, which is refined by modifying each global control. Global control parameters act as an abstraction layer of effect variables within the image processing algorithms applied in image-space (e.g., effect amounts, substrate attributes, atmospheric perspective).



(a) Normal *Viewport 2.0*

(b) Watercolor preset

(c) Oil paint preset

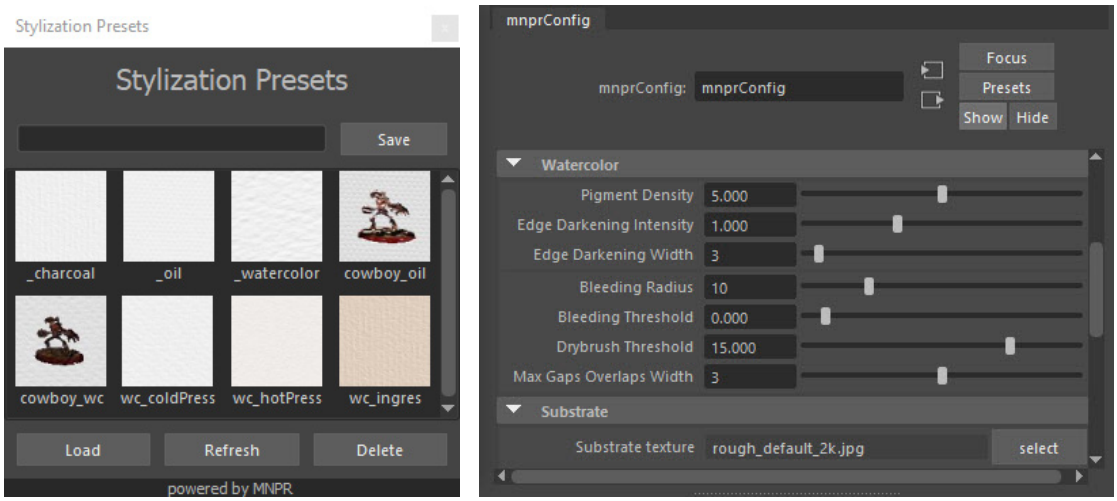
(d) Charcoal preset

Figure 6.4: Different style presets. Pony Cartoon model, CC BY Slava Zhuravlev.

To interact at these levels of control within the development framework (*Autodesk® Maya®*), two different different user interfaces had to be considered, which are briefly introduced next.

The style presets user interface (Figure 6.5a) had to be designed and developed from the ground up, mainly because the current *Maya* attribute presets framework does not have thumbnail views to help the user select the appropriate style. Essential features involve the ability to load, save and delete presets, as well as to visualize each preset within small thumbnails and a tool-tip with the global parameter values—which the preset would modify. The ability to refresh the presets library becomes useful when presets are modified outside of the user interface, allowing the tool to update its database.

The individual global parameters that are modified through the style presets are available in the *Attribute Editor* of the configuration node (Figure 6.5b). This user interface is automatically generated by *Maya*, depending on the attributes enabled in the stylization configuration node—set up by our custom plugin. We relied on this user interface, as it directly integrates with the plugin and *Maya* users are accustomed to working with it.



(a) Style presets window **(b)** Global control parameters in the *Attribute Editor*

Figure 6.5: User interfaces for high-levels of control.

6.1.2 Material Presets and Material Control

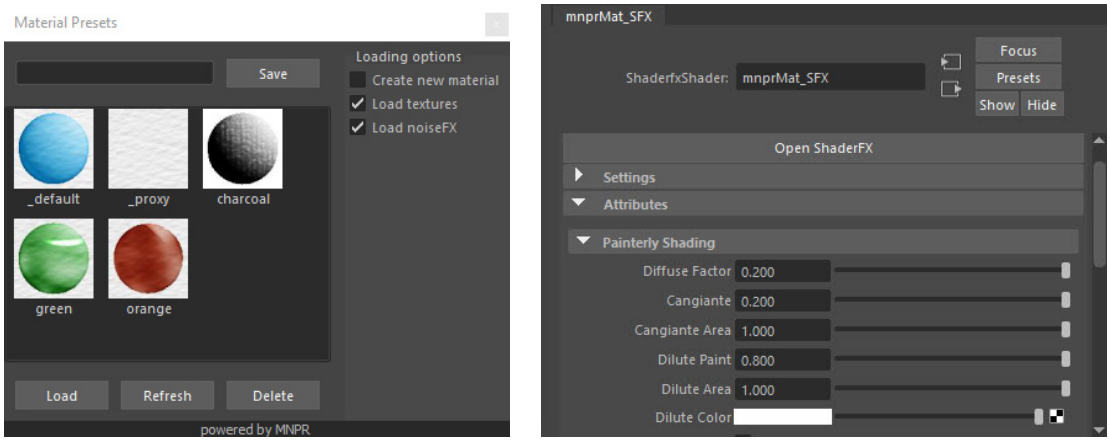
At the mid-level of control, material presets allow the artist to save or load predefined parameters of materials. These presets modify the underlying shader code and parameters of each *ShaderFX* material and enable the creation of material libraries with varying painterly shading properties (e.g., dilution, specularity, transparency). The material parameters can also be individually set within the material, providing individual control over each material property. By using the *ShaderFX* material framework within *Maya*, we could also offer a node-based interface to the artist, to extend or simplify the material, as they see fit. Additionally, the painterly materials are embedded with the ability to drive different image processing effects using procedural noise parameters N assigned in material-space, as shown in Figure 6.6. These allow natural-looking fluctuations in effects that might require them.

Figure 6.6: Procedural effects applied after the watercolor preset of Figure 6.4b.

There are three different tools, which need to be considered at this level of control. The material presets user interface (Figure 6.7a) was extrapolated from the style presets (Figure 6.5a) and extended with additional functionality. As there can be multiple materials assigned to 3D objects and multiple 3D objects can share a single material, an option to load a new material on the selected objects, instead of overriding an existing one, had to be included. The user also has the options to load the preset without the textures or the procedural control parameters saved with the material preset. These additional options complement the essential features elaborated for the style presets window (e.g., load, save, delete).

Each individual material parameter is available in the *Attribute Editor* of the each *ShaderFX* node (Figure 6.7b). This user interface and its sliders are automatically generated by the custom *ShaderFX* material, depending on how the *ShaderFX* graph is set up. Relying on the *ShaderFX* framework gives high-level shading control over the material through sliders, but also enables the visual development artists to have low-level shading control through the *ShaderFX* graph, with the freedom to modify any node within the shading network.

Finally, the procedural effects are controlled through a custom user interface (Figure 6.9a) that was designed to control each stylization effect, individually. Each effect is self-contained in its own section, which allows to define the intensity and scale of the noise through sliders, together with an offset (vertical slider). The noise can easily be restored to its default state with the reset button at the upper right corner. Additional options to toggle the noise effects on/off and the ability to switch between 2D/3D noise algorithms (*Simplex noise*) is also provided at the lower left of each section. These buttons and sliders are abstractions of node attributes in the *ShaderFX* graph, so the artist can modify effect parameters without having to open and navigate the node graph.



(a) Material presets window

(b) Material parameters in the *Attribute Editor*

Figure 6.7: User interfaces for mid-levels of control.

6.1.3 Mapped Control

At the lowest level of control, parameters assigned through mapped control M allow the artist to locally assign specific effects in object-space. This level of control provides the most refined control for the detail oriented artist, but it can also be the most time consuming. However, under real-time interactivity, these tools can be designed to assimilate a painting workflow, becoming immediately intuitive and fun for the artist to use. Mapped control possesses significant uses by itself, but a well thought-out combination of effects can also create targeted stylization concepts (e.g., watercolor turbulence using different pigment densities, as seen in Figure 6.8)—going beyond the specific effect that they were designed to produce.

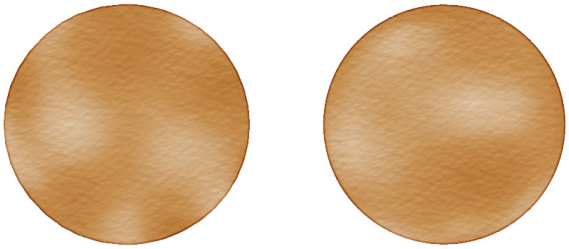
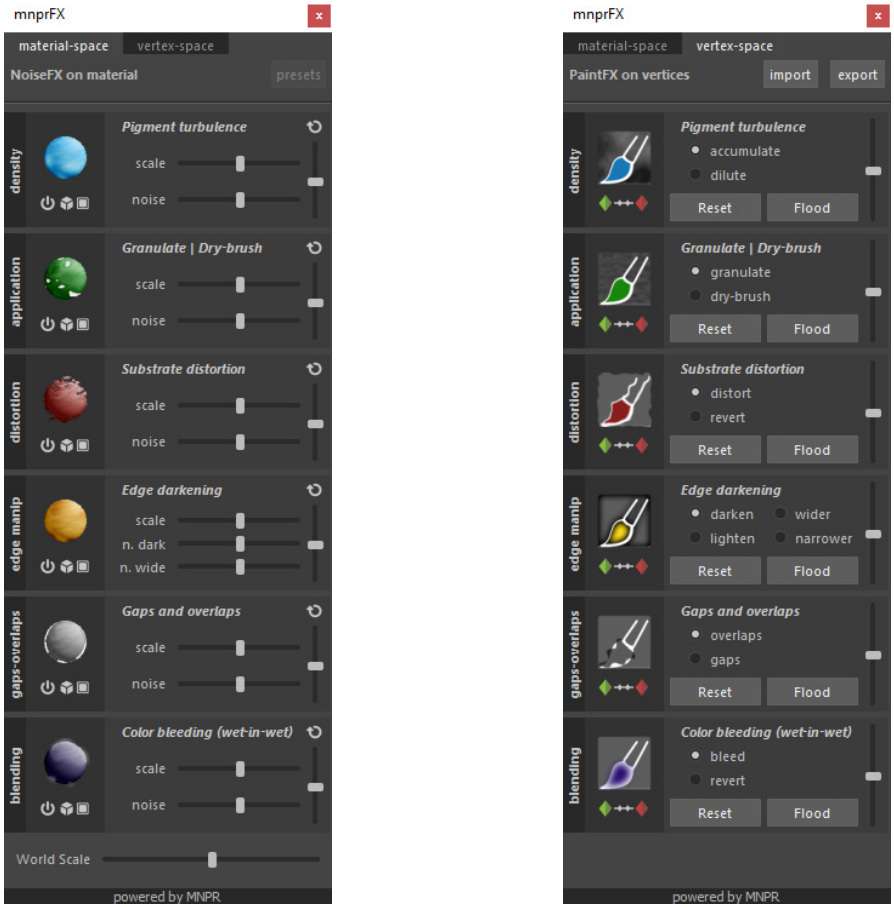


Figure 6.8: Left: Procedural pigment density (turbulence < 5 sec.). Right: Mapped pigment density (turbulence 40+ sec.).

The interaction at such a low level of control required a more complex user interface that could provide refined control over each effect. The low-level effect parameters in our implementation are assigned by painting vertex colors, with each channel corresponding to a particular effect. *Maya* offers a versatile vertex color painting tool, which earlier versions benefited from. However, as more effects and functionality were required, the need for a specialized user interface became necessary: *PaintFX*. The *PaintFX* user interface (Figure 6.9b) is an abstraction of *Maya*'s vertex color painting tool, which was designed to be more intuitive for stylization purposes and to include several additional functions.

Each effect has its own section (similarly to the *NoiseFX* interface), which contains the operation type (radio button) and the intensity of the effect parameter to be painted on (vertical slider). This automatically activates *Maya*'s vertex color tool within the appropriate vertex color set, channel and intensity (positive or negative). Flooding and resetting the painted parameters can also be easily controlled through the dedicated buttons within each effect. For animated control, buttons to keyframe the effect parameters (green), show the vertex color keyframes (gray) and delete existing keyframes (red) are provided underneath the brush icon. This considerably improves the otherwise tedious process of keying vertex color parameters using the *Maya* user interface.



(a) NoiseFX control window (mid-level)

(b) PaintFX control window (low level)

Figure 6.9: Interfaces to set effect parameters at the mid- and low-level of control.

6.1.4 Proxy Control

In parallel to most levels of control is proxy control, which comprises invisible 3D elements that only contribute to the render with localized stylization effect parameters P. However, by taking advantage of the 3D space they operate in, proxies encompass the assignment of stylization parameters from a high level of control—affecting the whole image by being placed in front of the camera, resembling NPR lenses [Neumann et al., 2007]—, to a low level of control—affecting a specific part in 3D space—, as visualized in Figure 6.10. This versatility is unique to proxies, since these operate as standalone elements in the scene, which can form arbitrary representations in object-space. Depending on their design, control proxies can support the previously mentioned procedural parameters in material-space and mapped parameters in object-space.

The proxy control we used was based on geometry, with a *ShaderFX* shader that would only render control parameters to the control targets. By doing so, we could take advantage of both, the *NoiseFX* and *PaintFX* user interfaces, to

assign procedural parameters and locally set parameters by painting on their proxy surface. In addition to this, the material provides a feathering attribute and the ability to map cookie masks to the proxy objects, which multiply whatever parameters it has by the mask values (similar to light cookies, useful for NPR lenses). These attributes are found in the *Attribute Editor*, as in Figure 6.7b.

Figure 6.10: Stylized result after mapped control, proxies are visualized in blue.

6.1.5 Results and Discussion

All these levels of control work in unison to produce the final stylized render. First, the 3D-space driven parameters, which include the procedural noise effect parameters N, the mapped effect parameters M and the proxy effect parameters P contribute to the image-space stylization maps M_{f_x}. These positive or negative parameters are combined at the rasterization stage $M_{f_x} = N + M + P$, as seen in Figure 6.11.

Once the rasterization stage is completed, the stylization pipeline modulates the effect parameters found in the stylization maps with the global effect parameters of the style during runtime. These parameters control the image processing stages, modifying the effect algorithms that are applied to the original rendered image, to finally create the stylized imagery, as shown in Figure 6.12.

We aimed to grant artists the widest range of control in object-space and, thereby, evaluate their preferences and uses at each level of control. For this purpose, we conducted a second user study, involving experienced artists and engineers, to evaluate the usefulness of each control and gather feedback to improve upon them.

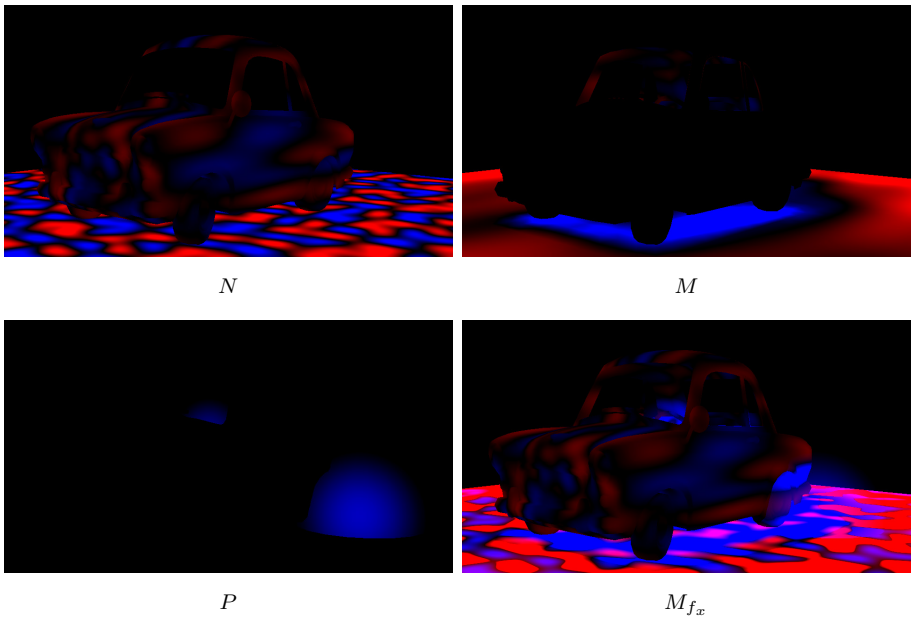


Figure 6.11: Effect parameters within the stylization map M_{f_x}. Positive values are visualized in blue, negative values in red.

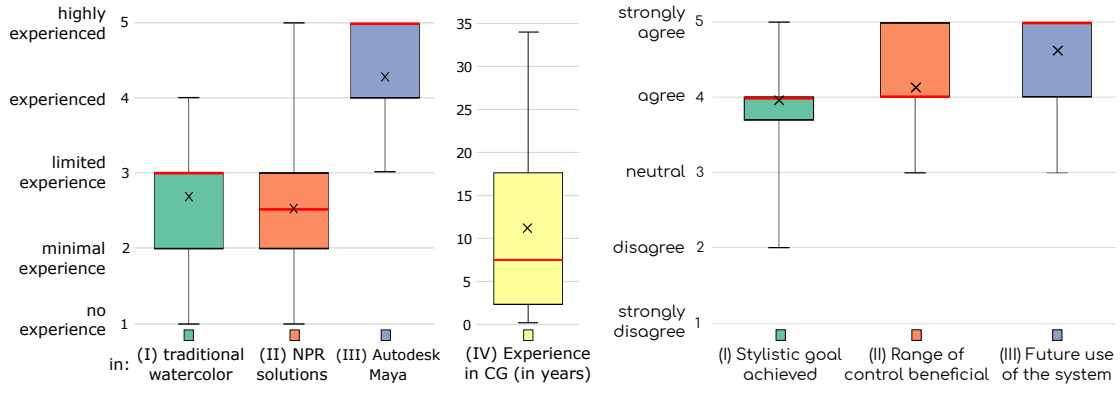
A total of 65 users expressed their interest to take part in our study and received our framework (MNPR) with a watercolor stylization pipeline. To bring everyone to the same level of proficiency using the framework and its tools, we included tutorials for each level of control and how to use them effectively. After a period of two weeks, 20 participants answered an online questionnaire that helped us gather their usage data and analyze their feedback. The individual responses are available within the Appendix A.

To assess their responses, we were initially interested in their prior experience (Figure 6.13a) using traditional watercolor, other NPR solutions and *Autodesk® Maya®*—together with their overall years of experience in a computer graphics related industry. The majority of users had limited to minimal experience using traditional watercolors (I), which is expected as we were targeting 3D artists and engineers. These results resembled their prior experience with other NPR solutions (II), probably reflecting on the scarce dedicated commercial NPR solutions available to date. Nonetheless, our users were mostly experienced with *Maya* (III) and within a wide spectrum of experience in computer graphics related industries (IV)—involving students and veterans alike.

To take part in the study, the participants were asked to commit to at least two hours when testing MNPR (our framework) and its distinct levels of control. However, their dedication exceeded our expectations, as most participants reported spending substantially more time testing our framework, as observed in Figure 6.14a. The additional time allowed them to familiarize themselves better with each level of control.

Figure 6.12: Final watercolor rendering leveraging all levels of control.

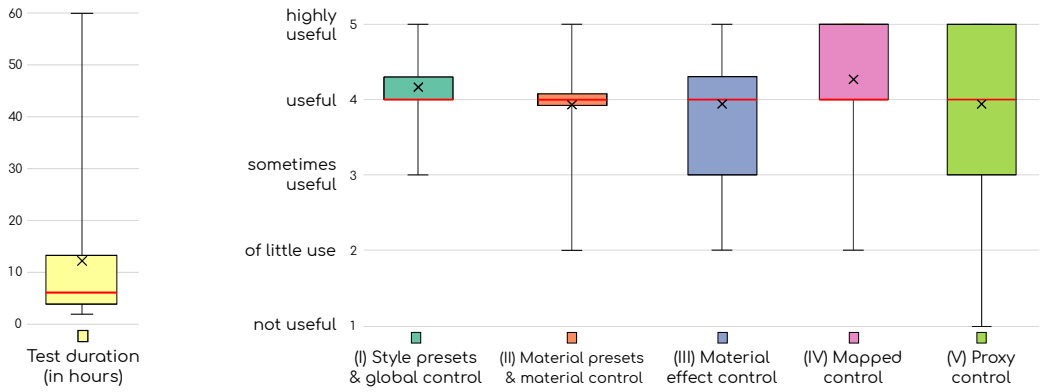
Regarding the levels of control (Figure 6.14b). Style presets and global control (I) were generally perceived as useful, tending towards highly useful in some cases. This result was expected, as these represent the main control over the stylized render and the starting point towards a stylized look. There was a general consensus that the material presets and material controls (II) were useful, with little deviation. We believe this result represents the general awareness of this level of control, especially considering existing photorealistic workflows that rely on material attributes to define the look of objects. Material effect controls (III), with their procedural noise parameters—providing mid-level control for effects—were generally perceived useful. This level of control took the participants the longest to understand, but most found its usefulness after experimenting with it. Mapped control (IV) was considered the most useful level of control by the majority of the participants. This is probably due to its localized nature and intuitive applica-



(a) Experience level of users

(b) Overall satisfaction

Figure 6.13: Box plots with prior experience and overall satisfaction of users.



(a) Duration **(b)** Levels of control rating

Figure 6.14: Results of the user experiment visualized within box plots.

tion through painting tools. Finally, proxy control (V), while being completely detached from the rendered subject, was generally found useful. This decoupled approach to art-direct the stylization received the most varied result, with some participants enthusiastically welcoming the addition (see Figure 6.15), while others not seeing their usefulness.

Regarding the overall satisfaction (Figure 6.13b), the participants tended to agree that they achieved their stylistic goal using our framework (I). We also asked the participants if they use other NPR solutions for their current workflow; and if they would have achieved their desired stylization faster using our framework. While they were mostly using offline workflows—which involve several stages with different software—their answers were mostly neutral, but with a tendency towards agreeing. We believe these responses were positive, considering that stylistic goals may differ widely between users and that they were constrained to only one stylization pipeline within our framework (watercolor), with limited learning time. We expect this to change when more stylization pipelines are developed and MNPR matures in the public domain. Of special importance to our study and our research is that most agree and even strongly agree that the range of control benefited their stylization (II), reinforcing our work and encouraging us to continue our pursuit towards art-directed tools. However, as one participant pointed out, it is important to find a balance within control, as this might distract users from the "bigger picture". Finally, an overwhelming majority of the participants strongly agreed that they would consider using our framework for their future projects (III).

Based on our experiments and reinforced by our user evaluation, we can conclude that the interaction spectrum, given by our different levels of control, was quite useful for artists—assisting the stylization process by enabling a faster and more versatile workflow. Additionally, since the interaction happens in real-time, the art-direction and rendering are performed concurrently, allowing the artists to see their changes immediately. We tested all these levels of control, working in

unison, using three distinct stylizations: watercolor, oil paint and charcoal. Our results are presented throughout Figure 6.16. Considering the experimental nature of these stylization pipelines, we have managed to produce satisfying results.

Figure 6.15: A screenshot of a scene by a test user where proxies (blue, left) are mostly used to drive the stylization (right).

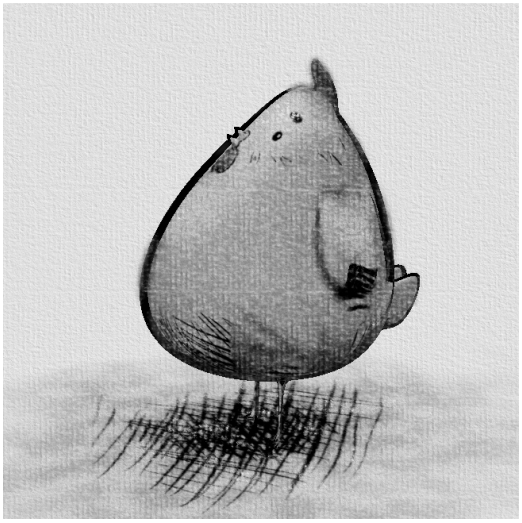
Limitations

While the developed tools to art-direct the stylization at different levels of control were generally found useful, these can be subject to further improvement and better alternatives. Based on the usefulness found by artists, we can especially argue that different ways to provide material and proxy control of effects could provide a promising area of future research, due to their rather unexplored nature and variance presented in their usefulness. An in-depth-study on human-computer interaction and a more refined user interface design would benefit the tools of the developed research prototype. However, such a study is outside the scope of my PhD studies and more suitable for researchers specialized in such fields.

An inherent limitation of our low-level art-direction worth highlighting lies on the vertex density and topology of polygonal objects. As parameters are assigned per-vertex, a certain degree of polygonal resolution is required to provide enough control. Mindful modeling and different levels of detail are suggested to overcome this limitation. Another alternative is to embed the control within object-space painted textures, this would allow for an arbitrary resolution of control, but would require properly unwrapped UVs for each geometric object. Finally, the use of *Ptex* [Burley and Lacewell, 2008] to drive the stylization could overcome this limitation, but would come at additional performance costs.

(a) Watercolor stylization. Baba Yaga's hut model, CC BY Inuciian.

(b) Oil paint stylization. Lizard Mage model, CC BY Marleen Vijgen, concept by Alex Braun.



(c) Charcoal stylization. Chicken Friend ;o model, CC BY MahoSway.

(d) Oil paint stylization. SIKA model, © Marine Quiviger.

Figure 6.16: Rendered frames using all levels of control in different styles.

6.2 Cross-stylistic Art-direction

Artistic visions can be represented in different styles, as seen in Figure 6.17. These artworks are an excellent example of two paintings using different media, where the art-direction remains mostly the same. Both masterfully portray a dynamic piece with the same subject in a mystical African setting.

The cross-stylistic art-direction found in traditional paintings is desirable in expressive NPR, as well. To make this possible, control semantics are necessary for the transfer of effects and these need to follow a defined control scheme for the expressive styles to work predictably. Based on our research, observations and discussions working with artists, we argued previously (Chapter 5) that stylization effects in watercolor can be generalized into four distinct categories:

- Pigment-based effects
- Substrate-based effects
- Edge-based effects
- Abstraction-based effects

These four categories can be extrapolated to work with other styles and used to directly correlate the stylization control parameters between them.

(a) Life-Spring, Graphite 4x12". © Dylan Scott Pierce.

(b) Life Spring, Watercolor 8x22". © Dylan Scott Pierce.

Figure 6.17: Paintings showing cross-stylization with different types of media.

6.2.1 Style Control Semantics and Scheme

The majority of art-directed parameters are in the stylization maps M_{f_x} within the framebuffer, of which each can manage an effect category. Within each effect category, there are semantics for the type of effect that can be modified. These semantics need to be sufficiently generic, yet semantically meaningful to be adapted to different styles and accepted by the NPR community. Additionally, these effects need to adhere to a control scheme (summarized in Table 6.1), that defines what semantics goes to which channel in the stylization maps—so that these can be interpreted by other styles. That way, stylization pipelines supporting the same effect categories, respecting the semantics and following the control scheme, would enable e.g., to map an art-directed rendering in a watercolor style (Figure 6.12) to an oil or charcoal style (Figure 6.18).

Figure 6.18: Cross-stylization visualized in oil paint (left) and charcoal (right) styles.

The stylization semantics I propose were based on the experience and observations that we have gathered while researching and iterating upon the watercolor stylization, and the implementation of the oil paint [Semmo et al., 2016] and charcoal [Chiew et al., 2018] stylizations. From developing and working on these different stylizations, we defined the following effects semantics.

Pigment-based effects

This category contains parameters that direct the way the pigmentation is rendered within a specific style:

- *Pigment variation.* Controls the degree at which the reflected color of a compound pigment varies towards one or another color. E.g., green pigmentation that deviates to a more blue or yellow color in certain parts.
- *Pigment application.* Controls how the pigment is placed over a substrate. This can be interpreted as the amount or pressure at which pigment is applied to achieve an effect. E.g., dry-brush application, thick application.

- *Pigment density.* Controls the concentration of the pigment placed over a substrate. This is especially relevant to transparent/translucent media (i.e., watercolor, ink, colored pencils), but can also influence opaque media. E.g., dilution, lightness, saturation.

Substrate-based effects

This category contains parameters that direct the way the substrate (i.e., canvas, paper, wall) affects a specific style:

- *Substrate distortion.* Controls the distortion caused by the substrate roughness on the rendered image. This is especially relevant for fluid media (i.e., watercolor, graffiti).
- *U- & V-inclinations.* Control the inclination of the substrate, which generally affects the direction at which patterns or marks from fluid media evolve. However, generalizing upon this, these parameters are used to define the offset of existing patterns or marks in a horizontal or vertical direction. E.g., bleeding direction, cross-hatching direction, stroke direction.

Edge-based effects

This category contains parameters that direct the way the edges of the subject are rendered within a specific style:

- *Edge intensity.* Controls the edge strength/intensity within the stylized render. E.g., linework darkness, edge darkening.
- *Edge width.* Controls the edge thickness of the stylized render. E.g., linework width, edge darkening width.
- *Edge transition.* Controls the edge transition of the subject in relation to neighboring elements. E.g., edge softening, gaps and overlaps.

Abstraction-based effects

This category contains parameters that direct the abstraction of the rendered subject within different styles.

- *Detail*. Controls the amount of detail at different parts of the stylized render within the subject. E.g., detail blur.
- *Shape*. Controls the amount of shape abstraction/distortion of the subjects. E.g., hand tremors.
- *Blending*. Controls the color blending at different parts of the stylized render. E.g., smudges, color bleeding.

By adhering to these semantics within the control scheme (Table 6.1), throughout the stylization pipeline, art-directed scenes can predictably change style and, for the most part, keep the intended effects and look of the expressive render (Figures 6.19 and 6.26).

Table 6.1: Stylization control scheme.

Channel	**Pigment** M_{f_x}	**Substrate** M_{f_x}	**Edge** M_{f_x}	**Abstraction** M_{f_x}
R	Variation	Distortion	Intensity	Detail
G	Application	U-inclination	Width	Shape
B	Density	V-inclination	Transition	Blending

6.2.2 Results and Discussion

To properly develop and evaluate the generalized control semantics and scheme, we had to consider other styles apart from watercolor. Therefore, we implemented the oil stylization of Semmo et al. [2015, 2016] (without color quantization) and invited a computer science undergraduate student (new to the field) to develop and implement a charcoal stylization as part of a final year project [Chiew et al., 2018].

Porting over Semmo et al.'s oil paint stylization work was quite straightforward and took approximately 2 weeks, which included re-writing the compute shaders to *HLSL* and redefining the order of the image processing operations. However, this time did not include the different development challenges that arose from porting a 2D stylization technique into 3D. A particular challenge was porting over the procedural impasto marks in a coherent way under animation. In this pursuit, we extended MNPR to include a way to extract the motion of each vertex by calculating the deltas with their previous position (after deformations) and made use of coherent noise [Kass and Pesare, 2011] to advect the procedural noise along

the object's motion. While the results are promising, calculating per vertex motion of fully deformable objects is slow and only enabled when real-time interaction is not required.

The charcoal stylization took understandably longer to develop, as it was not a full-time effort, there was no prior shader code and it involved a significant learning journey into image processing and shader development from the developer. Nonetheless, our framework managed to abstract the development process of a graphics engine, allowing to concentrate on the image processing operations and the interplay between them to create a charcoal stylization pipeline.

MNPR successfully supported both development efforts through the example code/documentation and allowed to easily embed the art-directed parameters within the image processing operations, following the control scheme at Table 6.1. Once these parameters were incorporated, the expressive control tools were managed by our framework, by simply defining the code to manipulate them. Both of these additional stylization pipelines are featured in our results, showcasing the successful adaptation of styles into MNPR, as well as the cross-stylistic support of art-directed effects. In general, the feedback from these development efforts was quite positive and allowed us to iterate on our framework, to improve upon it and extend its functionality.

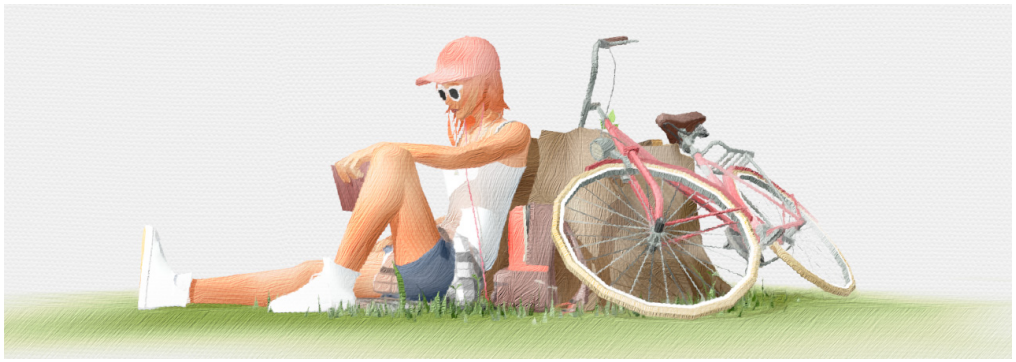
Once the oil paint and charcoal stylization were implemented, by generalizing our control semantics, our framework permitted the art-direction within one style to be visualized in others, as seen in Figures 6.19 and 6.26. These stylization pipelines all run within our framework (MNPR) in real-time and the user can interactively switch between them.

Limitations

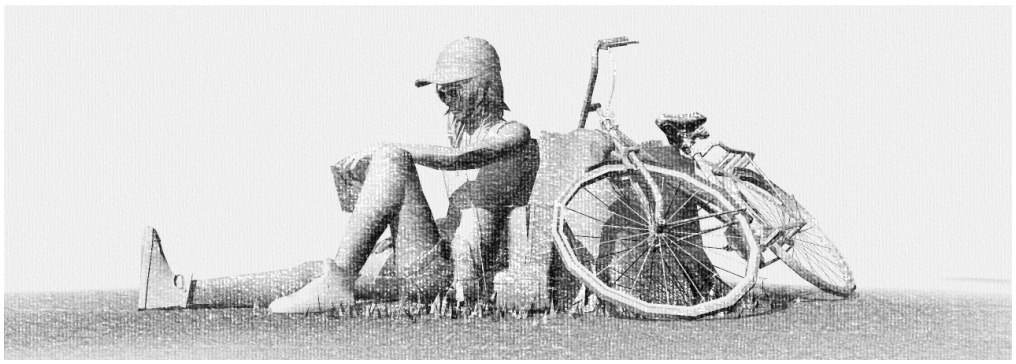
The cross-stylistic semantics that I propose enable art-direction in multiple styles in a predictive manner. However, although we have tested these style control semantics with watercolor, oil paint and charcoal with relative success, their applicability still needs to be further proven in other styles e.g., *hatching* [Suarez et al., 2016] or even *cubism* [Arpa et al., 2012]. Additionally, some styles require more than just a change in stylization pipelines and effects for the subject to acquire the intended look. A clear example is given when contrast in a specific style is achieved through color variations and another style through lightness values, as is the case with watercolor/oil and charcoal. As illustrated in Figure 6.20, a simple conversion from an oil paint stylization to a charcoal stylization will lack the contrast that defines the form of the subject. Notwithstanding, in this case, an adjustment to

Normal Viewport 2.0 render with painterly material

Watercolor stylization

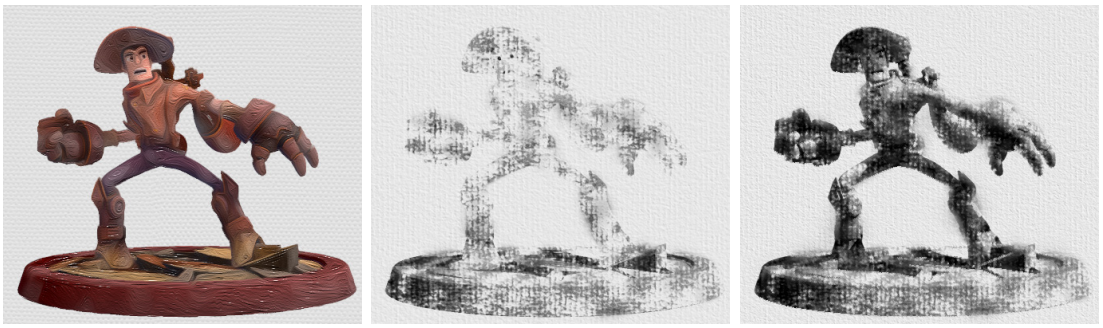


Oil paint stylization



Charcoal stylization

Figure 6.19: Art-directed cross-stylization. Summer Afternoon model, © Stevie Brown.



Oil paint Charcoal Material adjustment

Figure 6.20: Cross-stylization: oil paint to charcoal, Steam Cowboy model © Black Spire Studio.

the materials in the scene can let us modify the diffuse behavior (diffuse factor in the painterly material) to accommodate the look and bring the necessary contrast back (Figure 6.20c).

Most importantly, the proposed semantics might be subject to revisions and extensions. Being the first researcher formally proposing an art-directed cross-stylization framework in object-space, I only introduce the semantics as a generalized suggestion towards a common art-directed effects framework. Hopefully, these semantics can be adopted by the NPR community and continue evolving to a more mature state, where artists can visualize their creations in multiple styles with the same art-direction. While these semantics are neither final, nor applicable to all styles and effects, they provide a starting point to address cross-stylization paradigms in expressive rendering.

6.3 MNPR

The Maya Non-Photorealistic Rendering (MNPR) framework resulted as a product from my research in real-time watercolor rendering of 3D objects and animation with enhanced control. This development was required, as there was no expressive non-photorealistic rendering framework available in a real production software, which focused on enabling real-time art-direction throughout the interaction spectrum [Isenberg, 2016].

I specifically developed MNPR as a plugin for *Autodesk® Maya®*, as I wanted to make the watercolor stylization accessible for artists and existing projects, to evaluate and validate our research. The use of *Maya* propelled our toolset and allowed our research to reach a wide audience with varied assets, but these benefits came with closed-source caveats. The development insights of working with this popular digital content creation package, creating an art-directable pipeline with each level of control, together with the lessons learned, are elaborated in Section 6.3.1.

While I have explained our approaches to synthesize different effects of watercolor (Chapter 5), I have not elaborated on how these effects are set up within a watercolor stylization pipeline. Therefore, the watercolor stylization pipeline that I developed during my studies is introduced in Section 6.3.2, together with the steps to take for implementing other stylization pipelines withing MNPR. Any stylization pipeline developed in MNPR can take advantage of the custom art-directed tools for the different levels of control and, by following the proposed semantics and scheme, benefit from cross-stylization and a variety of pre-stylized assets.

Finally, this section and our contribution is concluded with a discussion on MNPR, its results, performance and possible future in the public domain.

6.3.1 Implementation within Autodesk® Maya®

To maximize the use of MNPR by artists and developers alike, the framework was implemented in one of the widely adopted digital content creation packages used in the animation and game industries, *Autodesk® Maya®*. Relying on this software allowed us to propel our toolset—by adopting existing tools and frameworks within *Maya*—and enabled the possibility to easily test the technology with artists. This approach was well received, as over 500 users worldwide registered as beta testers and could immediately get the stylization running with their assets.

While *Maya* is a closed-source, proprietary software, it is an "open product" that offers an extensive application programming interface (API) to extend the software. The *Maya API* is accessible through the *C++*, *Python* and *C#* languages. However, due to the need of high performance code and the available plugin examples, *C++* was chosen as the development language for MNPR.

The plugin, originally based on the *viewRenderOverride* plugin available in the *Maya Devkit*, quickly evolved with each iteration, up to the point where it became unsustainable and tedious to extend, as each change required hard-coded changes in many parts of the source code. Based on this prior experience working with render overrides and the *Maya Viewport 2.0 API*, I decided to develop a new render override plugin, extending some API classes and methods to design and abstract a modular framework. This new plugin focused on being easily extensible and generic enough to support other stylization pipelines. The result of this effort was MNPR. While I will not go into the details of how MNPR was programmed, a class diagram can be found in the Appendix A (Figure A.1), together with links to the source code, for interested developers.

The shaders for the stylization pipeline were initially written in *HLSL* for the *DirectX 11* viewport, as the support for custom *OpenGL* render overrides was, initially, not mature enough within *Maya*. As the *Viewport 2.0 API* matured and requests from users for a *Mac* version increased, we started translating the shaders to *GLSL*, but kept developing shaders in *HLSL* first, as debugging *GLSL* shaders within *Maya* is still a frustrating endeavor—error logs are often not available.

While the back-end of the stylization pipeline was set up in *C++* and either *HLSL* or *GLSL* depending on the platform, the front-end of MNPR was coded in *Python* and *Qt*, more specifically *PySide2*. The communication between the *Python* tools and the plugin was accomplished either via a custom *mnpr* command or by modifying attributes in a custom configuration node, depending on the required usage. Attributes that are meant to change regularly were handled by the configuration node (i.e., engine settings, effect parameters), whereas settings that should not change dynamically were handled by the *mnpr* command (i.e., enable/disable operations, refresh shaders).

Installation and user interaction

The installation of MNPR within *Maya* was designed to be as straightforward and non-invasive as possible. MNPR comes in a self-contained folder with all its required files. This folder can be placed in any directory and the installation is accomplished by dragging and dropping an installation file into the *Maya* viewport.

This file then runs in the background and makes any necessary changes, adding the MNPR folder to the *Maya* environment paths. After restarting *Maya*, MNPR will be fully installed, showing all the relevant tools through a custom MNPR shelf (Figure 6.21). This custom shelf contains all the required stylization tools, divided into three sections.

Figure 6.21: MNPR shelf in *Maya*.

Starting from the left, the first section is the most important, providing a direct button to the MNPR online documentation, followed by each level of control, from highest to lowest: style presets [style] and global control [conf] (Section 6.1.1); material presets [mPre], material control [mat] and NoiseFX [nFx] (Section 6.1.2); and mapped control aka. PaintFX [pFx] (Section 6.1.3).

These stylization tools are followed by a custom viewport renderer [rendr] (Figure 6.22). The renderer allows to easily save still images of arbitrary resolutions (max. 16384 pixels in any dimension due to hardware limitations) or to render a video (playblast) with the style. It will automatically set the engine settings of MNPR accordingly and disable operations that are not required in the final image (i.e., heads-up display).

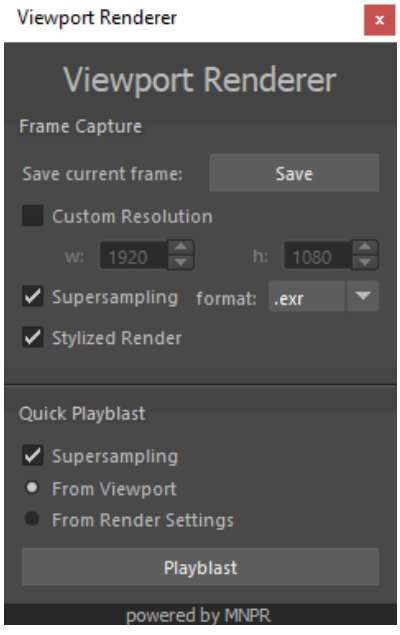


Figure 6.22: Viewport renderer window.

The second section on the MNPR shelf (Figure 6.21) contains all three supported lights: ambient, directional and spot light. These shelf buttons are followed by a test scene [test] that automatically creates a sphere with basic lighting to check

if the stylization is working correctly. The stylization can then be debugged using the custom operation breakdown [pass] user interface (Figure 6.23). The operation breakdown is the most useful tool to debug stylization pipelines within MNPR. It contains a list of all the operations of the current stylization, together with a checkbox to enable/disable these and a button to reload the shaders, to the left of each operation. Reloading shaders forces MNPR to load the corresponding shader again and show any changes performed in the shader code. Each render target, stored in the framebuffer, can be interactively displayed in the viewport by setting the active render target. The active target can then be troubleshooted using the debugging utilities at the bottom of the user interface. Each specific channel/s (RGBA) can be isolated and the images may switch between color spaces interactively. As negative parameters cannot normally be visualized, these can be inverted for visual debugging, as well. The last shelf button of the second section opens the project's web repository to provide feedback regarding any bugs or future request.

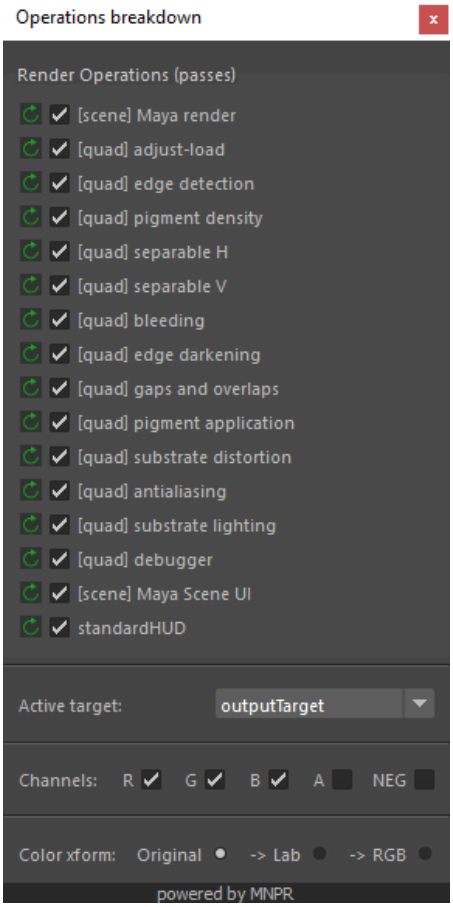


Figure 6.23: Operation breakdown window.

The last section of the shelf (Figure 6.21) is used strictly for development purposes. The first shelf button will refresh all shaders in the current stylization pipeline. The final shelf button will toggle the plugin within *Maya*, making it easier to unload the plugin, to compile a new version and to reload the new version within the application.

Tools for different levels of control

Within MNPR, broken down in the schematic of Figure 6.24, the 3D scene interaction is entirely managed by *Maya*. However, enabling the different levels of control for our stylization pipeline required us to develop multiple custom tools, which were previously introduced in Section 6.1. While enabling global controls using the *Maya* node attributes is straightforward, making the other levels of control required special considerations.

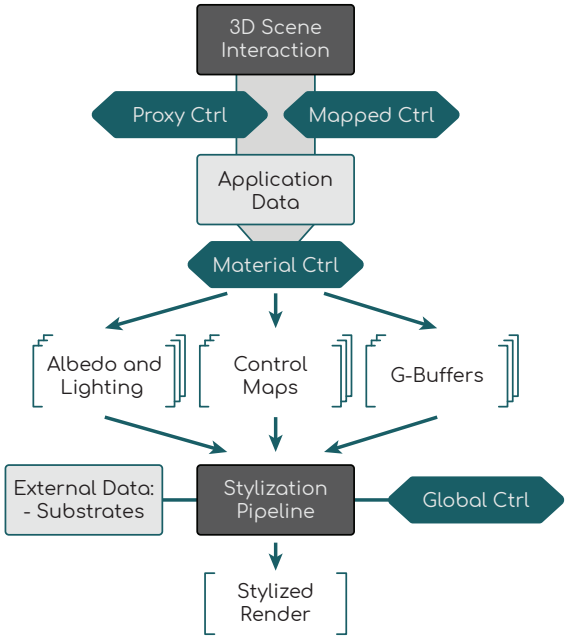


Figure 6.24: Schematic of MNPR with its control stages within the 3D pipeline.

In 3D space, we handle the creation of proxy controls from any mesh object, by first disabling the casting and receiving of shadows so that these are not present in the shadow map calculation. We then assign a custom proxy material (using the *ShaderFX* framework), which will only render the proxy object to the control maps, once the application data is rasterized. A proxy material preset, loaded from our custom material presets tool, automatically performs these changes, so that the user is only concerned with the art-direction of the "invisible" object.

Any mesh object in the 3D scene can be embedded with mapped control, even proxies. We handle mapped control using vertex color sets, which are created and managed automatically by our custom *PaintFX* tool. The *PaintFX* tool modifies three RGBA vertex color sets that render out the controls for the four effect categories that we previously defined (Section 6.2). By using vertex colors (instead of textures, for instance), we have a sparse control representation that can further take advantage of Maya's animation capabilities to keyframe effect parameters.

Mapped parameters can only be rendered from the application data if the material supports them. Our custom real-time *ShaderFX* material renders the 3D application data into the different render targets required by our stylization pipeline: the albedo—which includes some painterly shading properties—, diffuse and specular targets, any necessary geometry data (*G-buffers*) and the four stylization control maps. *Maya*'s *ShaderFX* is a real-time, node-based shader editor that gives us the flexibility to recompile the GPU shader on demand, depending on the operating system and the material properties we require. This mechanism saves unnecessary computation time of optional shading requirements—such as specularity, transparency and procedural noise parameters.

The material control of effects is directed by procedural noise parameters that are also embedded into the *ShaderFX* material. Our custom *NoiseFX* tool allows the artist to control noise functions from an abstracted interface, avoiding the navigation through each shader graph towards the specific noise parameter in a node. Our materials accumulate the control contributions generated by these noise functions and the previously mapped parameters, found in meshes and proxies, into the stylization control maps—following the scheme found in Table 6.1.

Lessons learned

While working with *Autodesk® Maya®* has propelled the set of features and the pool of artists that we can access, developing for this closed-source software has its caveats. I would like to share a brief list of lessons learned, for any developer or researcher looking into using and expanding MNPR.

- Documentation, example code and user-base of the *Viewport 2.0 API* are limited, but support in the *Autodesk* CG community (AREA) can be helpful.
- *Maya* will not return certain shader errors when working with *GLSL* shaders. This led us to develop in *HLSL* first and then port to *GLSL* afterwards.
- Keep hardware limitations in mind when rendering to multiple render targets. A fragment shader can render to a maximum of 8 render targets.
- Packing control parameters into 32 bit floating point targets will not support a simple accumulation of parameters from different levels of control.
- *Maya*'s internal texture painting tool will not work on custom materials, which led us to map to vertex colors instead.

I hope that by sharing these development insights, together with the source code (Appendix A), future developers will be motivated to implement stylization pipelines within *Maya*, benefit from the toolset that I have developed during my studies, and create technologies that artists can immediately use.

6.3.2 Stylization Pipeline

Setting up stylization pipelines within MNPR is straightforward, as it has been simplified and abstracted for people who are not accustomed to working with the *Maya API*. To adapt a stylization pipeline into MNPR and take advantage of the art-directed tools, three steps are required:

- define and create attributes for global effect parameters (*C++*);
- outline the custom stylization pipeline (*C++*);
- define *NoiseFX* and *PaintFX* controls (*Python*).

Once these steps have been implemented, the plugin can be compiled and the stylization pipeline can be run inside of *Maya*. More details on how exactly these three steps are performed can be found at the *readme* file within the repository of the source code (Appendix A).

Depending on the style that is being sought after, the order of image processing operations might differ within the stylization pipeline. While I have previously introduced the synthesis of characteristic watercolor effects (Chapter 5), I have not elaborated on how exactly the watercolor stylization pipeline was set up.

Watercolor stylization pipeline

The synthesis of characteristic watercolor effects needs to work in unison to generate a successful watercolor depiction. Therefore, a thoughtful stylization pipeline is required, which will take the relationship between effects into account.

The stylization pipeline begins once all the required render targets have been either rendered by the painterly material or loaded by MNPR, see Figure 6.24. With all the rendered data a series of image processing operations, driven by shaders (*HLSL* or *GLSL*), are defined and run. These operations conform to the stylization pipeline (Figure 6.25), which is further driven by the global control parameters.

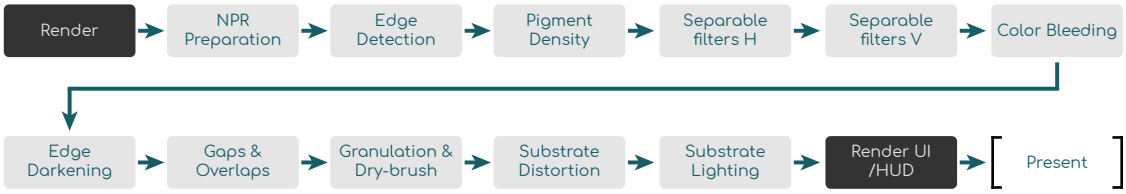


Figure 6.25: Schematic of the stylization pipeline with its different shader operations.

From left to right, the first render stage rasterizes the 3D geometry into multiple render targets, including the color image, lighting contributions, z-buffer, and control masks with the effect parameters. During the NPR preparation stage, the z-buffer is transformed into linear depth and the lighting is applied onto the color image. Then, the edge detection runs the RGBD *Sobel* filter, followed by the pigment density stage, which concentrates color or dilutes them at different parts of the image according to their assigned turbulent density. The separable filter stages perform edge blurs for edge darkening, extended edges for gaps & overlaps, and blurs the image for later bleeding. Afterwards, the color bleeding stage blends part of the blurred image to produce localized bleeding. Additionally, the control parameter masks related to edge-based effects are also modified, as darkened edges and gaps & overlaps should not appear in these bled areas.

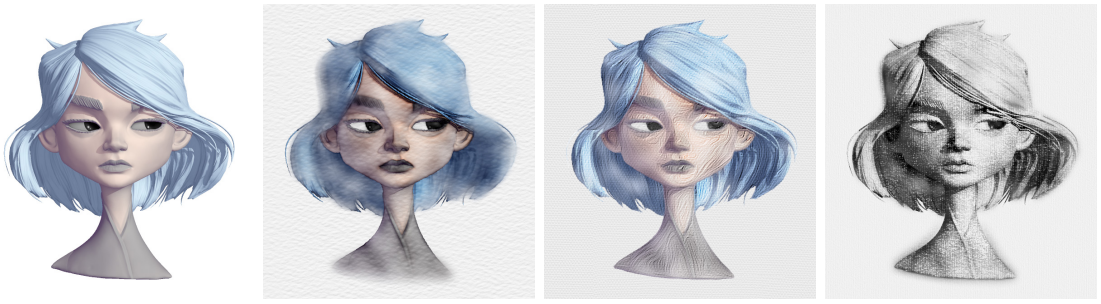
At the second row of Figure 6.25, the edge- and substrate-based effects are processed. Edge-based effects begin at the edge darkening stage, concentrating the pigmentation at the previously blurred edges. Gaps & overlaps use the previously extended edges to find their neighboring colors and either mix with them or produce gaps. The substrate-based effects begin at the granulation and dry-brush stage, deriving each effect from the acquired heightmap. Then, the substrate distortion modifies the UVs according to the substrate normals, shifting the sampled color. Once all these effects have been computed, the deferred substrate lighting is performed using the substrate normals. Finally, the remaining user-interface/heads-up-display is rendered and the final watercolorized result is presented to the user. Almost all these stages make extensive use of the control parameter masks, enabling localized art-directed effects in object-space that retain their spatial coherence. I encourage the interested reader to inspect the source code (Appendix A) for more detailed insights between each operation.

Limitations

The real-time implementation of the watercolor stylization does not present significant limitations apart from some directional artifacts within the separable filters—due to performance optimizations. However, the parallel nature of shaders requires special consideration compared to offline approaches, as only the currently evaluated pixel can be altered at each stage. The development of stylization pipelines within MNPR could also have benefited from a modular, node-based approach. This would enable developers to iterate quicker and alleviate the complexity of managing multiple rendering targets and several rendering stages only through code. Unfortunately, due to time constraints and the research focus of my studies, this development was not pursued.

6.3.3 Results and Discussion

We have presented MNPR, a framework for expressive non-photorealistic rendering of 3D computer graphics, developed as a plugin to *Autodesk® Maya®*. The focus of the framework was to power art-directed stylization pipelines throughout the interaction spectrum, optimizing the workflow of users with different preferences and skill levels. The development of the framework was iterated through the constant feedback of its users and two separate user studies (Sections 5.5 and 6.1.5), which helped refine the software and the user interaction. Currently, MNPR has been open-sourced with the permissive *MIT License*, in the hopes of facilitating future research and development of art-directed stylization pipelines and motivating its adoption in the *Expressive* community—as a practical answer to Isenberg's 2016 call towards designing the interaction with, or for algorithmic support. The source code can be found at the institutional data repository of the *Nanyang Technological University* [Montesdeoca, 2018] and at the project's *Git* repository (Appendix A).



normal, viewport 2.0 | watercolor | oil paint | charcoal

Figure 6.26: Cross-stylization of an art-directed character. Miss Seagull model, © Julien Kaspar.

MNPR empowers our stylization pipelines in watercolor, oil and charcoal, but can easily be extended by developers due to its cross-stylistic and open-source nature. Stylized results in all these styles have been presented through Figures 6.12, 6.16, 6.18 to 6.20, 6.26 and 6.27.

The individual performances of each stylization pipeline are measured in Table 6.2 with the following configuration: *NVIDIA® GeForce® GTX 1080*, *Intel® Xeon® E5-2609* @2.4Ghz and 16Gb of RAM using *Autodesk® Maya® 2018*. The developed framework, without any custom stylization pipeline, involves five different passes to render, adjust or load data, provide channel or color-space debugging utilities and perform anti-aliasing. The most complex pipeline, with 25 different image processing passes (including 6 separable filters) still manages to run at over 60 frames per second when rendering scenes without baked animated geometry caches. The benchmark in Figure 6.27 is affected by the alembic cache of the animated scene, which performs at 37fps with the normal *Maya Viewport 2.0*.

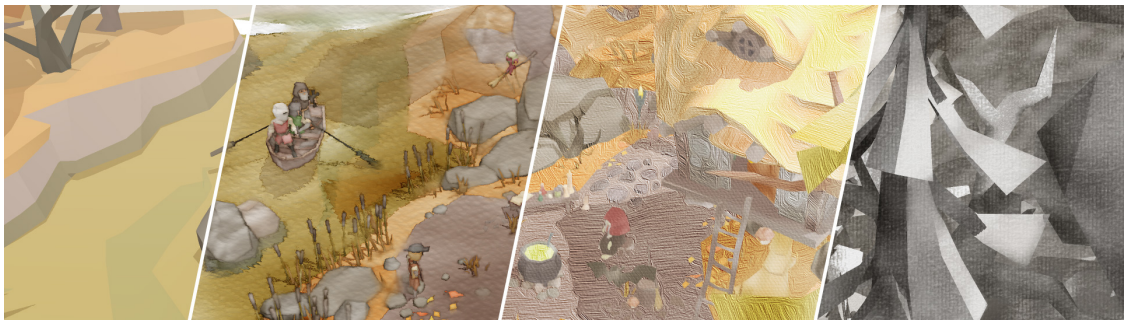


Figure 6.27: Art-directed cross-stylization of a complex environment. Baba Yaga's hut model, CC BY Inuciian.

Table 6.2: Performance in frames per second (fps) for the different styles of Figure 6.26 (288.800 triangles), Figure 6.27 (54.140 triangles) and Figure 6.19 (14.450 triangles) with a screen resolution of 1920 × 1080 pixels.

Style	Passes	Figure 6.19	Figure 6.26	Figure 6.27
Framework	5	305	275	32
Watercolor	13	170	179	27
Oil paint	25	85	84	26
Charcoal	17	108	102	27

Future work

While MNPR provides a useful starting point towards bringing NPR research to the hands of artists, it still remains to be tested and refined with other styles and in production. For example, we have not implemented a method to retain the motion coherence of substrate-based effects yet, as this is still an open problem that needs to be convincingly solved [Bousseau et al., 2006; Bénard et al., 2009, 2011; Hays and Essa, 2004]. A production-ready solution would need to take this into account if temporal continuity is desired i.e., when alternating substrate patterns—which happen naturally with traditionally painted animation that present substrate-based effects—are not desired. We have also shared some limitations through the lessons learned within our implementation insights (Section 6.3.1). Some of these limitations pertain to the current available hardware, but others pertain to *Autodesk Maya* and its closed-source nature, from which we hope to spur support in the future and work together towards having more expressive NPR available to artists and productions alike.

7. Conclusions

This thesis presented the research contributions that pushed the state-of-the-art towards a more faithful watercolor synthesis in object-space. Not only did the work presented herein introduce novel and improved approaches towards recreating characteristic effects of watercolor, but also embedded real-time art-direction throughout the interaction spectrum. This feat allowed the creation of direct stylization pipelines that validated and facilitated the artistic interaction with the rendered style and the iterative art-direction process—which was extrapolated with generalized style control semantics, that allow cross-stylization in expressive non-photorealistic rendering.

In an increasingly animated world, where most animated productions resort to computer generated imagery, I hope that my work can contribute towards using the powerful hardware we possess to reminisce, revisit and reinvent the wide gamut of styles available in painted art, within an animated context. In the same way that scientists and artisans empowered artists to fully express themselves in color and detail, we, as researchers and developers, are now responsible to concoct new colors and craft new tools to enable 3D artists to realize their visions and contribute to renewed ways of visual storytelling.

There is much more to computer graphics than photorealism and there is an immense and exciting world in everyone's minds that needs to be told. It is time to explore visual alternatives and create the technology to empower the future generations of visual storytellers.

> *Now that 'perfect' design is possible with the click of a mouse, the industrialized world has become nostalgic for 'imperfect' design. As computer-aided everything takes over our lives we begin to realize, little by little, what is missing from the high-tech world. We realize that a crooked line sometimes has more soul than a perfectly straight one...*
> (David Byrne [2003])

7.1 Summary of Contributions

The contributions of the work conducted during my PhD studies can be broken down into three distinctive groups, which altogether answered the research questions proposed in Section 1.2.

A real-time painterly material for 3D objects was introduced (Chapter 4), which proposed an alternate way on how light is reflected from objects, bringing colors into the foreground of shading control. The material contains customizable shading parameters that allow artists to override the diffuse factor and the shading color. Additionally, the shading within the light and shadow parts of objects can be individually defined and tinted towards any desired color. Thereby also supporting the use of dilution and cangiante illumination commonly found in painted watercolors. Thus, the custom material is intended to replicate a more painterly thought process, while still keeping support for texture mapping techniques and reflectance models from common real-time materials. More importantly, the material has the embedded ability to drive and render the effect parameters that were either locally assigned on the vertices or procedurally applied through noise algorithms within the material. Apart from defining the rendered color and driving the effects, the material outputs a series of additional object-space data into multiple render targets to help the stylization process.

The watercolor stylization is the main contribution of my work and focused on synthesizing characteristic effects, essential to the traditional watercolor medium (Chapter 5), in real-time. The synthesis is done via image processing within a stylization pipeline, taking advantage of the rendered 3D data, provided by the painterly materials. The stylization concentrates on emulating characteristic pigment-, edge-, substrate- and abstraction-based effects that, when used in unison, can produce a watercolorized image. The palette of watercolor effects in object-space was extended by introducing novel approaches towards color bleeding, gaps & overlaps and dry-brush, together with the dilution and cangiante illumination from the painterly watercolor material. Additionally, existing approaches were enhanced and improved upon, by synthesizing a robust edge darkening with gradual pigment accumulation, efficient and depth-aware substrate distortion, accurate substrate granulation and controllable substrate lighting. All these effects are complemented by real-time art-directed local and procedural control in object-space, thereby retaining their spatial coherence. A user study involving professional 3D artists validated our watercolor synthesis and our direct stylization pipeline.

The direct stylization pipeline is finally introduced, which is the framework that enabled expressive control over the watercolor stylization—covering the interaction spectrum [Isenberg, 2016]—and proposed style control semantics to allow

cross-stylization into other media (Chapter 6). From a high level of control using style presets and global control parameters, to a medium level of control using material presets and procedural control parameters, down to a low-level of control with mapped control parameters. These levels are complemented by proxy control, which decouples the parameters from the subject, by using invisible control proxies that can further take advantage of effects applied at other levels of control. Each level of control was then evaluated and validated through a second user study involving a wide gamut of 3D artists. All the art-direction tools drive the characteristic effects of the watercolor stylization in real-time by following a proposed control scheme, governed by the developed style control semantics. The style control semantics and the control scheme are the first of its kind within object-space approaches that enabled the possibility of cross-stylization, demonstrated by transferring the art-direction to other non-photorealistic styles, such as oil painting and charcoal drawing. The contribution of this thesis concluded with a thorough discussion about the implementation of a direct stylization framework within the popular digital content creation software, Autodesk® Maya®and the final release of MNPR (the developed framework) into the public domain under the permissive *MIT license*.

7.2 Future Work

> *I knew by 1991'92, that the computer was gonna be doing tremendous amounts of things for us. Let's not kid ourselves about that, but I don't want it to look that way. I want it to look like somehow there are brushstrokes in everything, and somebody made this lovingly by hand all alone in their basement somewhere.* (Roy E. Disney, [Disney, 2000])

The work towards a truly faithful synthesis of watercolors in animation is not likely to finish anytime soon. The synthesis still has many limitations that need to be addressed, compared to its real counterpart. Such is the case of watercolor effects that were not considered in our approaches e.g., pigment clumping/separation, pigment fractals, hue variations, salt application. These effects provide exciting new avenues for future research, together with improvements over existing approaches within the current synthesis—as the proposed approaches still present limitations, elaborated in Section 5.5.2. Existing approaches might benefit from alternatives that make more extensive use over 3D data to enable the further refinement of the effects e.g., projecting effects in cast shadows, directional bleeding and pigment turbulence follow-through/settle-back under animation. Additionally, further improvements over the acquisition of edge and substrate data will surely help the current approaches towards synthesizing edge- and substrate-based effects.

In traditionally painted watercolor animation, the substrate changes with each painted frame, presenting flickering on effects that depend on the substrate, which can become strenuous to watch over time. Synthesized substrate-based effects are also prone to flicker, because of this. While this is normal within traditional media, researchers have devised approaches to reduce this by proposing the use of dynamic substrates [Bousseau et al., 2006; Bénard et al., 2009, 2011; Hays and Essa, 2004]. Unfortunately, existing approaches have not convincingly solved this problem, still presenting temporal coherence problems and traces of the 'shower door' effect. With the development of coherent noise [Kass and Pesare, 2011] and high performant noise synthesis based on textures [Heitz and Neyret, 2018], there might be a possibility to solve this issue in the near future.

MNPR, the art-directed tools and the style control semantics will need to be refined and improved under real production settings. By open sourcing its development, I hope to motivate its usage and attract contributions from external developers, studios and *Autodesk*® itself. We can all benefit from a mature, open-source framework that makes expressive research more accessible, enabling entirely new visuals and storytelling for artists, validating the methods of NPR researchers and inspiring new contributions from the research community.

A promising area of future work concerns direct stylization pipelines and their use in animated productions. GPU hardware has only recently reached the maturity to support entire stylization pipelines in real-time, which brings the opportunity to process complex image processing operations in milliseconds. While working with *Autodesk® Maya®* helped the development of the research prototype used throughout this thesis, it presented limitations that were impossible to circumvent because of the closed source nature of this commercial application. An external, software agnostic renderer, which could be bound to any real-time viewport within a digital content creation (DCC) package would be an excellent NPR solution, which would allow any stylization and its art-directed tools throughout various software packages—with a consistent real-time look across applications.

Bibliography

Abgott A. 2007. *Daring Color: Mix and Mingle Watercolor on Your Paper*. F+W Media. https://books.google.com/books?id=-4VBWPZYk08C

Akenine-Moller T., Haines E., Hoffman N., Pesce A., Iwanicki M., and Hillaire S. 2018. *Real-Time Rendering* (4rd ed.). A K Peters/CRC Press.

Arpa S., Bulbul A., Capin T., and Ozguc B. 2012. Perceptual 3D Rendering Based on Principles of Analytical Cubism. *Computers & Graphics* 36, 8 (2012), 991 – 1004. DOI:https://doi.org/10.1016/j.cag.2012.06.003

Bacher H. 2007. *Dream Worlds: Production Design for Animation*. Focal Press. http://books.google.com/books?id=BKljePzKBvQC

Barbagallo R. 2003. A Blade of Grass. (2003). http://animationartconservation.com/a-blade-of-grass.html

Barmpoutis A., Bozia E., and Wagman R. S. 2010. A Novel Framework for 3D Reconstruction and Analysis of Ancient Inscriptions. *Machine Vision and Applications* 21, 6 (2010), 989–998. DOI:https://doi.org/10.1007/s00138-009-0198-7

Birren F. 1987. *Principles of Color: A Review of Past Tradition and Modern Theories*. Schiffer Publishing. http://books.google.com/books?id=rgeSuAAACAAJ

Blinn J. F. 1977. Models of Light Reflection for Computer Synthesized Pictures. *SIGGRAPH Comput. Graph.* 11, 2 (July 1977), 192–198. DOI:https://doi.org/10.1145/965141.563893

Bousseau A., Kaplan M., Thollot J., and Sillion F. X. 2006. Interactive Watercolor Rendering with Temporal Coherence and Abstraction. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering*. ACM, 141–149. DOI:https://doi.org/10.1145/1124728.1124751

Bousseau A., Neyret F., Thollot J., and Salesin D. 2007. Video Watercolorization Using Bidirectional Texture Advection. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26, 3 (2007), 104. DOI:https://doi.org/10.1145/1276377.1276507

Brooks L. 1977. *The Many Ways of Water & Color: Watercolor, Acrylic, Casein, Gouache, Inks, Mixed Techniques*. Westport, Conn.: North Light Publishers. https://books.google.com/books?id=9SgRAQAAMAAJ

Brown C. W. 2007. *The Watercolor Flower Artist's Bible: An Essential Reference for the Practicing Artist*. Chartwell Books. https://books.google.com/books?id=ZRe6GQAACAAJ

Burgess J., Wyvill G., and King S. A. 2005. A System for Real-Time Watercolour Rendering. In *Computer Graphics International*. IEEE, 234–240. DOI:https://doi.org/10.1109/CGI.2005.1500426

Burley B. and Lacewell D. 2008. Ptex: Per-Face Texture Mapping for Production Rendering. *Computer Graphics Forum* (2008). DOI:https://doi.org/10.1111/j.1467-8659.2008.01253.x

Byrne D. 2003. When Bad Art is Good. (2003). https://www.utne.com/community/when-bad-art-is-good

Bénard P., Bousseau A., and Thollot J. 2009. Dynamic Solid Textures for Real-Time Coherent Stylization. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*. ACM, 121–127. DOI:https://doi.org/10.1145/1507149.1507169

Bénard P., Bousseau A., and Thollot J. 2011. State-of-the-Art Report on Temporal Coherence for Stylized Animations. *Computer Graphics Forum* 30, 8 (2011), 2367–2386. DOI:https://doi.org/10.1111/j.1467-8659.2011.02075.x

Bénard P., Cole F., Kass M., Mordatch I., Hegarty J., Senn M. S., Fleischer K., Pesare D., and Breeden K. 2013. Stylizing Animation By Example. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* 32, 4 (2013), 119. DOI:https://doi.org/10.1145/2461912.2461929

Cabral B. and Leedom L. C. 1993. Imaging Vector Fields Using Line Integral Convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*. ACM, 263–270. DOI:https://doi.org/10.1145/166117.166151

Cartwright J. P. 2015. *Mastering Watercolors: A Practical Guide*. [San Bernardino, CA: CreateSpace Independent Publishing Platform, 2015]. https://books.google.com/books?id=gKCdAQAACAAJ

Chen J., Turk G., and MacIntyre B. 2008. Watercolor Inspired Non-photorealistic Rendering for Augmented Reality. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*. ACM. DOI:https://doi.org/10.1145/1450579.1450629

Chen Z., Kim B., Ito D., and Wang H. 2015. Wetbrush: GPU-Based 3D Painting Simulation at the Bristle Level. *ACM Trans. Graph.* 34, 6, Article 200 (Oct. 2015), 200:1–200:11 pages. DOI:https://doi.org/10.1145/2816795.2818066

Chiew Y. X., Seah H. S., and Montesdeoca S. E. 2018. Real-Time Art-Directed Charcoal Cyber Arts. In *2018 International Conference on Cyberworlds (CW)*. IEEE.

Chu N. S. H. and Tai C.-L. 2005. MoXi: Real-Time Ink Dispersion in Absorbent Paper. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)* 24, 3 (2005), 504–511. DOI:https://doi.org/10.1145/1073204.1073221

Cignoni P., Montani C., Scopigno R., and Rocchini C. 1998. A General Method for Preserving Attribute Values on Simplified Meshes. In *Proceedings of the Conference on Visualization '98 (VIS '98)*. IEEE Computer Society Press, 59–66. http://dl.acm.org.ezlibproxy1.ntu.edu/citation.cfm?id=288216.288224

Cohen J., Olano M., and Manocha D. 1998. Appearance-preserving Simplification. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. ACM, 115–122. DOI:https://doi.org/10.1145/280814.280832

Comaniciu D. and Meer P. 1999. Mean Shift Analysis and Applications. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, Vol. 2. 1197–1203. DOI:https://doi.org/10.1109/ICCV.1999.790416

Cunzi M., Thollot J., Paris S., Debunne G., Gascuel J.-D., and Durand F. 2003. Dynamic Canvas for Immersive Non-photorealistic Walkthroughs. In *Proceedings Graphics Interface (Graphics Interface)*. A K Peters, LTD., 121–130. https://hal.inria.fr/inria-00510176

Curtis C. 2015. Fishing, by David Gainey (1999). (2015). https://vimeo.com/118042611

Curtis C. J., Anderson S. E., Seims J. E., Fleischer K. W., and Salesin D. H. 1997. Computer-Generated Watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1997)*. ACM Press/Addison-Wesley Publishing Co., 421–430. DOI:https://doi.org/10.1145/258734.258896

Curtis D. and Capon R. 2005. *Light and Mood in Watercolour*. Batsford. https://books.google.com/books?id=x8HNXyFePRkC

Disney W. 2000. The Fantasia Legacy: Fantasia Continued. (Nov. 2000). https://www.imdb.com/title/tt0407780/ Bonus material of Fantasia 2000.

DiVerdi S., Krishnaswamy A., Měch R., and Ito D. 2013. Painting with Polygons: A Procedural Watercolor Engine. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (2013), 723–735. DOI:https://doi.org/10.1109/TVCG.2012.295

Durand F. 2002. An Invitation to Discuss Computer Depiction. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering (NPAR '02)*. ACM, 111–124. DOI:https://doi.org/10.1145/508530.508550

Dutcher L. 2013. *Watercolor: Paintings by Contemporary Artists*. Chronicle Books. https://books.google.com/books?id=QkhcDpQAg6IC&lpg=PP1

Edin R. and Jepsen D. 2010. *Color Harmonies: Paint Watercolors Filled with Light*. F+W Media. https://books.google.com/books?id=_oRUp4d7RRwC

Ferwerda J. A. 2003. Three Varieties of Realism in Computer Graphics. *Proc.SPIE* 5007 (2003), 5007 – 5007 – 8. DOI:https://doi.org/10.1117/12.473899

Fišer J., Jamriška O., Lukáč M., Shechtman E., Asente P., Lu J., and Sýkora D. 2016. Stylit: Illumination-Guided Example-Based Stylization of 3D Renderings. *ACM Trans. Graph.* 35, 4, Article 92 (July 2016), 92:1–92:11 pages. DOI:https://doi.org/10.1145/2897824.2925948

Fišer J., Lukáč M., Jamriška O., Čadík M., Gingold Y., Asente P., and Sýkora D. 2014. Color Me Noisy: Example-Based Rendering of Hand-Colored Animations With Temporal Noise Control. *Computer Graphics Forum* 33, 4 (2014), 1–10. DOI:https://doi.org/10.1111/cgf.12407

Franks G. 1988. *Watercolor: Drybrush Technique*. Walter Foster Pub. https://books.google.com/books?id=tiS0v-5Guv8C

Fritz B. 2013. THQ Bankruptcy Auction Closes; Video Game Rivals Pick Up Assets. (2013). http://articles.latimes.com/print/2013/jan/23/business/la-fi-ct-thq-auction-20130124

Gatys L. A., Ecker A. S., and Bethge M. 2016. Image Style Transfer Using Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2414–2423. DOI:https://doi.org/10.1109/CVPR.2016.265

Gooch A., Gooch B., Shirley P., and Cohen E. 1998. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 447–452. DOI:https://doi.org/10.1145/280814.280950

Good O. 2016. Crytek Employees Say They're Not Being Paid, Again. (2016). https://www.polygon.com/2016/12/10/13908156/crytek-employees-not-paid

Gossett N. and Chen B. 2004. Paint Inspired Color Mixing and Compositing for Visualization. In *IEEE Symposium on Information Visualization*. 113–118. DOI: https://doi.org/10.1109/INFVIS.2004.52

Gwynn K. 2004. *Painting in Watercolor*. Eagle Editions. https://books.google.com/books?id=qoEWAAAACAAJ

Haase C. S. and Meyer G. W. 1992. Modeling pigmented materials for realistic image synthesis. *ACM Trans. Graph.* 11, 4 (1992), 305–335. DOI:https://doi.org/10.1145/146443.146452

Hall P. and Lehmann A.-S. 2013. *Don't Measure—Appreciate! NPR Seen Through the Prism of Art History*. Computational Imaging and Vision, Vol. 42. Springer-Verlag London, Book 16, 333–351. DOI:https://doi.org/10.1007/978-1-4471-4519-6

Hargraves M., Wilcox S., for British Art Y. C., Collection P. M., and of Fine Arts V. M. 2007. *Great British Watercolors: From the Paul Mellon Collection at the Yale Center for British Art*. Yale Center for British Art. https://books.google.com/books?id=S70MEDiMMpUC

Harvill A. 2007. *Effective Toon-Style Rendering Control Using Scalar Fields*. Memo. http://graphics.pixar.com/library/ToonRendering/index.html

Hays J. and Essa I. 2004. Image and Video Based Painterly Animation. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering (NPAR '04)*. ACM, 113–120. DOI:https://doi.org/10.1145/987657.987676

Heitz E. and Neyret F. 2018. High-Performance By-Example Noise using a Histogram-Preserving Blending Operator. *Proceedings of the ACM on Computer Graphics and Interactive Techniques (issue for the ACM SIGGRAPH / Eurographics Symposium on High-Performance Graphics 2018)* 1, 2 (Aug. 2018), 25. DOI:https://doi.org/10.1145/3233304

Hertzmann A. 2002. Fast Paint Texture. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering (NPAR '02)*. ACM, 91–ff. DOI:https://doi.org/10.1145/508530.508546

Hoffmann T. 2012. *Watercolor Painting: A Comprehensive Approach to Mastering the Medium*. Watson-Guptill Publications. http://books.google.com/books?id=mXpYAcfVd-wC

Hopewell B., John Lang. 2018. Why Animation Is the Hottest Genre at Cannes' Market. (2018). http://variety.com/2018/film/festivals/animation-cannes-fireheart-missing-link-1202805624/

Isenberg T. 2013. *Evaluating and Validating Non-photorealistic and Illustrative Rendering*. Springer London, Chapter 15, 311–331. DOI:https://doi.org/10.1007/978-1-4471-4519-6_15

Isenberg T. 2016. Interactive NPAR: What Type of Tools Should We Create?. In *NPAR'16 Proceedings of the Workshop on Non-photorealistic Animation and Rendering*. The Eurographics Association, 89–96. DOI:https://doi.org/10.2312/exp.20161067

Janson K. 2003. *The DC Comics Guide to Inking Comics*. Watson-Guptill.

Johan H., Hashimoto R., and Nishita T. 2004. Creating Watercolor Style Images Taking Into Account Painting Techniques. *The Journal of the Society for Art and Science* 3, 4 (2004), 207–215. DOI:https://doi.org/10.3756/artsci.3.207

Kalnins R. D., Markosian L., Meier B. J., Kowalski M. A., Lee J. C., Davidson P. L., Webb M., Hughes J. F., and Finkelstein A. 2002. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)* 21 Issue 3 (2002), 755–762. DOI:https://doi.org/10.1145/566654.566648

Kaplan M. and Cohen E. 2005. A Generative Model for Dynamic Canvas Motion. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*. Eurographics Association, 49–56. DOI:https://doi.org/10.2312/compaesth/compaesth05/049-056

Kass M. and Pesare D. 2011. Coherent Noise for Non-photorealistic Rendering. *ACM Trans. Graph.* 30, 4, Article 30 (2011), 30:1–30:6 pages. DOI:https://doi.org/10.1145/2010324.1964925

Kelts R. 2016. CG Gains a 'Real' Foothold in Anime. (2016). https://www.japantimes.co.jp/culture/2016/02/20/general/cg-gains-real-foothold-anime/

Kessenich J., Sellers G., and Shreiner D. 2016. *OpenGL®Programming Guide: The Official Guide to Learning OpenGL®, Version 4.5 with SPIR-V* (9 ed.). Addison-Wesley Professional. https://books.google.com/books?id=u_M-jwEACAAJ

Lake A., Marshall C., Harris M., and Blackstein M. 2000. Stylized Rendering Techniques for Scalable Real-Time 3D Animation. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering (NPAR '00)*. ACM, 13–20. DOI:https://doi.org/10.1145/340916.340918

Lambert J. H. 1760. *Photometria*. https://books.google.com/books?id=tbM6AAAAcAAJ

Lei S. I. E. and Chang C.-F. 2004. Real-Time Rendering of Watercolor Effects for Virtual Environments. In *Advances in Multimedia Information Processing - PCM 2004: 5th Pacific Rim Conference on Multimedia*, Kiyoharu Aizawa, Yuichi Nakamura, and Shin'ichi Satoh (Eds.). Springer Berlin Heidelberg, 474–481. DOI:https://doi.org/10.1007/978-3-540-30543-9_60

Lent J. A. 2001. *Animation in Asia and the Pacific*. Indiana University Press. http://books.google.com/books?id=TNJulkqR774C

Lent J. A. and Ying X. 2013. Chinese Animation: An Historical and Contemporary Analysis. *Journal of Asian Pacific Communication* 23, 1 (2013), 19–40. DOI:https://doi.org/10.1075/japc.23.1.03len

Lu J., Barnes C., DiVerdi S., and Finkelstein A. 2013. RealBrush: Painting with Examples of Physical Media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* 32, 4 (2013), 1–12. DOI:https://doi.org/10.1145/2461912.2461998

Luft T. and Deussen O. 2005. Interactive Watercolor Animations. In *Pacific Conference on Computer Graphics and Applications*. http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-27549

Luft T. and Deussen O. 2006a. Real-Time Watercolor for Animation. *Journal of Computer Science and Technology* 21, 2 (2006), 159–165. DOI:https://doi.org/10.1007/s11390-006-0159-9

Luft T. and Deussen O. 2006b. Real-Time Watercolor Illustrations of Plants Using a Blurred Depth Test. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering*. ACM, 11–20. DOI:https://doi.org/10.1145/1124728.1124732

Luft T., Kobs F., Zinser W., and Deussen O. 2008. Watercolor Illustrations of CAD Data. In *Computational Aesthetics in Graphics, Visualization, and Imaging*, Douglas W. Cunningham, Victoria Interrante, Paul Brown, and Jon McCormack (Eds.). The Eurographics Association. DOI:https://doi.org/10.2312/COMPAESTH/COMPAESTH08/057-063

Lum E. B. and Ma K.-L. 2001. Non-photorealistic Rendering Using Watercolor Inspired Textures and Illumination. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*. 322–330. DOI:https://doi.org/10.1109/PCCGA.2001.962888

Lü C. and Chen X. 2014. Image Watercolorization Based on Visual Weight-Map. In *2014 7th International Congress on Image and Signal Processing (CISP)*. 233–237. DOI:https://doi.org/10.1109/CISP.2014.7003783

Martín D., Arroyo G., Rodríguez A., and Isenberg T. 2017. A Survey of Digital Stippling. *Computers & Graphics* (2017), 24–44. DOI:https://doi.org/10.1016/j.cag.2017.05.001

Miki A. and Satou H. 2015. Isao Takahata and His Tale of the Princess Kaguya. (2015). https://www.imdb.com/title/tt6478412/

Ming-Ming C., Guo-Xin Z., Mitra N. J., Xiaolei H., and Shi-Min H. 2011. Global Contrast Based Salient Region Detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 409–416. DOI:https://doi.org/10.1109/CVPR.2011.5995344

Mitchell J., Francke M., and Eng D. 2007. Illustrative Rendering in Team Fortress 2. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering (NPAR '07)*. ACM, 71–76. DOI:https://doi.org/10.1145/1274871.1274883

Montesdeoca S. E. 2018. MNPR. (2018). DOI:https://doi.org/10.21979/N9/KU4B6S

Morris B. 2010. *Fashion Illustrator*. Laurence King Publishing. https://books.google.com/books?id=R0WSQQAACAAJ

Ness M. 2016. Experiments in Animation: Disney's Fantasia 2000. Macmillan. (Sept. 2016). https://www.tor.com/2016/09/29/experiments-in-animation-disneys-fantasia-2000/

Neumann P., Isenberg T., and Carpendale S. 2007. NPR Lenses: Interactive Tools for Non-photorealistic Line Drawings. In *Smart Graphics*, Andreas Butz, Brian Fisher, Antonio Krüger, Patrick Olivier, and Shigeru Owada (Eds.). Springer Berlin Heidelberg, 10–22. DOI:https://doi.org/10.1007/978-3-540-73214-3_2

Nienhaus J., M. Doellner. 2003. Edge-Enhancement: An Algorithm for Real-Time. *Journal of WSCG* 11, 1 (2003). http://hdl.handle.net/11025/1689

O'Conor K. 2017. GPU Performance for Game Artists. (2017). http://www.fragmentbuffer.com/gpu-performance-for-game-artists/

Ong K. S. 2003. *Mastering Light & Shade in Watercolor*. International Artist. http://books.google.com/books?id=1DlZNwAACAAJ

Papari G. and Petkov N. 2011. Edge and Line Oriented Contour Detection: State of the Art. *Image and Vision Computing* 29, 2–3 (2011), 79 – 103. DOI:https://doi.org/10.1016/j.imavis.2010.08.009

Parramón's Editorial Team . 2000. *All about Techniques in Color*. Barron's Educational Series. https://books.google.com/books?id=tiUKAAAACAAJ

Parrish K. 2018. Steam Saw More than 7,500 Titles Added to Its Library throughout 2017. (2018). https://www.digitaltrends.com/gaming/steam-library-grows-over-7500-titles-2017/

Perlin K. 1985. An Image Synthesizer. *SIGGRAPH Comput. Graph.* 19, 3 (July 1985), 287–296. DOI:https://doi.org/10.1145/325165.325247

Perlin K. 2002. Standard for Perlin Noise. (2002). https://patents.google.com/patent/US6867776

Pharr M., Jakob W., and Humphreys G. 2016. *Physically Based Rendering: From Theory to Implementation*. Elsevier Science. https://books.google.com/books?id=iNMVBQAAQBAJ

Phong B. T. 1975. Illumination for Computer Generated Pictures. *Commun. ACM* 18, 6 (1975), 311–317. DOI:https://doi.org/10.1145/360825.360839

Pollard J. G. 2013. *Watercolor Unleashed: New Directions for Traditional Painting Techniques*. Cincinnati, Ohio: North Light, [2013]. First edition. http://books.google.com/books?id=PDQGHOhL42gC

Quilez I. 2016. Quill: VR Drawing in the Production of Oculus Story Studio's New Movie. In *ACM SIGGRAPH 2016 Real-Time Live! (SIGGRAPH '16)*. ACM, Article 41, 41:26–41:26 pages. DOI:https://doi.org/10.1145/2933540.2933549

Read P. and Meyer M.-P. 2000. *Restoration of Motion Picture Film*. Elsevier Science. http://books.google.com/books?id=ai_ORUt8VlwC

Reinhard E., Ashikhmin M., Gooch B., and Shirley P. 2001. Color Transfer Between Images. *IEEE Comput. Graph. Appl.* 21, 5 (2001), 34–41. DOI:https://doi.org/10.1109/38.946629

Saito T. and Takahashi T. 1990. Comprehensible Rendering of 3-D Shapes. *SIGGRAPH Comput. Graph.* 24, 4 (Sept. 1990), 197–206. DOI:https://doi.org/10.1145/97880.97901

Saitzyk S. 1987. *Art Hardware: The Definitive Guide to Artists' Materials*. Watson-Guptill. https://books.google.com/books?id=iC6BQgAACAAJ

Salesin D. H. 2002. Non-photorealistic Animation & Rendering: 7 Grand Challenges. Keynote talk at Second International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002, Annecy, France, June 3–5, 2002). (June 2002). http://www.research.microsoft.com/~salesin/NPAR.ppt

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

Schmid J., Senn M. S., Gross M., and Sumner R. W. 2011. OverCoat: An Implicit Canvas for 3D Painting. *ACM Trans. Graph.* 30, 4, Article 28 (July 2011), 28:1–28:10 pages. DOI:https://doi.org/10.1145/2010324.1964923

Schmidt T.-W., Pellacini F., Nowrouzezahrai D., Jarosz W., and Dachsbacher C. 2016. State of the Art in Artistic Editing of Appearance, Lighting and Material. *Computer Graphics Forum* 35, 1 (2016), 216–233. DOI:https://doi.org/10.1111/cgf.12721

Schultz W. 2018. What Is a AAA Video Game? ThoughtCo. (June 2018). thoughtco.com/what-is-aaa-game-1393920.

Semmo A. and Döllner J. 2015. Interactive Image Filtering for Level-of-Abstraction Texturing of Virtual 3D Scenes. *Computers & Graphics* 52 (2015), 181 – 198. DOI:https://doi.org/10.1016/j.cag.2015.02.001

Semmo A., Dürschmid T., Trapp M., Klingbeil M., Döllner J., and Pasewaldt S. 2016. Interactive Image Filtering With Multiple Levels-of-Control on Mobile Devices. In *Proceedings of the ACM SIGGRAPH Asia Symposium on Mobile Graphics and Interactive Applications*. ACM, NA. DOI:https://doi.org/10.1145/2999508.2999521

Semmo A., Limberger D., Kyprianidis J. E., and Döllner J. 2015. Image Stylization by Oil Paint Filtering Using Color Palettes. In *Proceedings of the Workshop on Computational Aesthetics*. Eurographics Association, 149–158. DOI:https://doi.org/10.2312/exp.20151188

Semmo A., Limberger D., Kyprianidis J. E., and Döllner J. 2016. Image Stylization by Interactive Oil Paint Filtering. *Computers & Graphics* 55 (2016), 157 – 171. DOI:https://doi.org/10.1016/j.cag.2015.12.001

Shanes E., Joll E., Turner J., and of Arts (Great Britain) R. A. 2000. *Turner: The Great Watercolours*. Royal Academy of Arts. https://books.google.com/books?id=fMvqAAAAMAAJ

Shanghai Animation Film Studio . 2003. Chinese Classic Animation: Te Wei Collection. (2003). http://ezlibproxy1.ntu.edu/login?url=http:

//search.ebscohost.com/login.aspx?direct=true&db=cat00103a&AN=ntu.a501492&site=eds-live&scope=site

Shukla U. 2015. The Girl Who Cried Flowers: A Short Film. (2015). http://auryn.ink/?p=41

Small D. 1991. Simulating Watercolor by Modeling Diffusion, Pigment, and Paper Fibers. *SPIE Proceedings* 1460, Image Handling and Reproduction Systems Integration (1991), 140–146. DOI:https://doi.org/10.1117/12.44417

Smith A. and Ardill T. 2011. *Watercolour*. London: Tate; New York: Distributed in the United States by Harry N. Abrams, 2011. https://books.google.com/books?id=rJeCcAAACAAJ

Smith M. 2002. Part 2, A Tribute to John Lasseter. (2002). http://www.laughingplace.com/News-ID112270.asp

Smith R. 2003. *The Artist's Handbook*. DK Pub. https://books.google.com/books?id=14RUAAAAMAAJ

Soan H. 2014. *The Artist's Color Guide - Watercolor: Understanding Palette, Pigments and Properties*. F&W Media, Incorporated. https://books.google.com/books?id=GsAUngEACAAJ

Spencer J. 2017. Love It or Hate It, We're Stuck With 3D & CG Work in Anime. (2017). https://jonspencerreviews.wordpress.com/2017/01/12/cg-in-anime-a-brief-history-discussion/

Suarez J., Belhadj F., and Boyer V. 2016. Real-Time 3D Rendering with Hatching. *The Visual Computer* (2016), 1–16. DOI:https://doi.org/10.1007/s00371-016-1222-3

Taylor J. 2003. *Watercolour Wisdom: Lessons From a Lifetime of Painting and Teaching*. Newton Abbot: David & Charles, 2003. https://books.google.com/books?id=a9D-PAAACAAJ

Tedeschi M., Dahm K., Walsh J., and Huang K. 2008. *Watercolors by Winslow Homer: The Color of Light*. Chicago: The Art Institute of Chicago; New Haven: Yale University Press, c2008. 1st ed. https://books.google.com/books?id=332CDQAAQBAJ

Todo H., Anjyo K., Baxter W., and Igarashi T. 2007. Locally Controllable Stylized Shading. *ACM Trans. Graph.* 26, 3, Article 17 (July 2007). DOI:https://doi.org/10.1145/1276377.1276399

Todo H., Anjyo K., and Yokoyama S. 2013. Lit-Sphere Extension for Artistic Rendering. *The Visual Computer* 29, 6 (01 Jun 2013), 473–480. DOI:https://doi.org/10.1007/s00371-013-0811-7

Tomasi C. and Manduchi R. 1998. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 839–846. DOI:https://doi.org/10.1109/ICCV.1998.710815

Van Laerhoven T., Liesenborgs J., and Van Reeth F. 2004. Real-Time Watercolor Painting on a Distributed Paper Model. In *Computer Graphics International*. IEEE, 640–643. DOI:https://doi.org/10.1109/CGI.2004.1309281

Van Laerhoven T. and Van Reeth F. 2005. Real-Time Simulation of Watery Paint. *Computer Animation and Virtual Worlds* 16, 3-4 (2005), 429–439. DOI:https://doi.org/10.1002/cav.95

Vanderhaeghe D., Vergne R., Barla P., and Baxter W. 2011. Dynamic Stylized Shading Primitives. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-photorealistic Animation and Rendering (NPAR '11)*. ACM, 99–104. DOI:https://doi.org/10.1145/2024676.2024693

Verrier R. 2013. Are Hollywood Studios Cranking out Too Many Animated Films? (2013). http://articles.latimes.com/2013/aug/20/business/la-fi-ct-animation-20130820

Verrier R. 2015. DreamWorks Animation to Cut 500 Jobs. (2015). http://www.latimes.com/entertainment/envelope/cotown/la-et-ct-dreamworks-layoffs-20150122-story.html

Wakabayashi H. C. 2002. *Lilo & Stitch: Collected Stories From the Film's Creators* (1st ed.). Disney Editions. https://books.google.com/books?id=9dtkAAAAMAAJ

Wang M., Wang B., Fei Y., Qian K., Wang W., Chen J., and Yong J.-H. 2014. Towards Photo Watercolorization with Artistic Verisimilitude. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1451–1460. DOI:https://doi.org/10.1109/TVCG.2014.2303984

Wang N. 2008. Water-and-Ink Animation. (2008). http://usa.chinadaily.com.cn/culture/2008-06/04/content_11848479_2.htm

Ward G. J. 1992. Measuring and Modeling Anisotropic Reflection. *SIGGRAPH Comput. Graph.* 26, 2 (July 1992), 265–272. DOI:https://doi.org/10.1145/142920.134078

Williams L. 1978. Casting Curved Shadows on Curved Surfaces. *SIGGRAPH Comput. Graph.* 12, 3 (Aug. 1978), 270–274. DOI:https://doi.org/10.1145/965139.807402

Winnemöller H. 2013. NPR in the Wild. In *Image and Video-Based Artistic Stylisation*, Paul Rosin and John Collomosse (Eds.). Computational Imaging

and Vision, Vol. 42. Springer-Verlag London, Book 17, 353–372. DOI:https://doi.org/10.1007/978-1-4471-4519-6

Worley S. 1996. A Cellular Texture Basis Function. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, 291–294. DOI:https://doi.org/10.1145/237170.237267

Worth L. and Deutch Y. 1982. *Painting in Watercolours*. Tunbridge Wells, Kent: Search Press, c1982, 1991.

Wyvill G. 1999. The Nature of Noise. (1999). http://www.cs.otago.ac.nz/graphics/Geoff/NoisePages/Nature.html

You M., Jang T., Cha S., Kim J., and Noh J. 2013. Realistic Paint Simulation Based on Fluidity, Diffusion, and Absorption. *Computer Animation and Virtual Worlds* 24, 3-4 (2013), 297–306. DOI:https://doi.org/10.1002/cav.1500

Zhu Y. and Rosen S. 2010. *Art, Politics, and Commerce in Chinese Cinema*. Hong Kong: Hong Kong University Press. https://books.google.com/books?id=DUqmtfTNZ8UC

A. Appendix

This Appendix compiles supplementary material relevant to the conducted research on real-time watercolor rendering of 3D objects and animation with enhanced control.

The source code of MNPR, the expressive non-photorealistic framework developed as a byproduct of the conducted research, is first given in Appendix A.1. It is complemented with a class diagram that contains the most important classes within MNPR and its methods, to aid in the visualization of the source code and the understanding of the different elements of the software.

The raw data from the two conducted user studies is also included in Appendix A.2, for future researchers that wish to do their own interpretation or follow through with additional studies. This research data includes both, the questionnaires and the collected responses.

All 3D models presented in this thesis were used with permission in each individual publication where these were featured. The *Creative Commons* models were provided by *Sketchfab*. The terms of the *Creative Commons Attribution 4.0 International License* are found at: https://creativecommons.org/.

A.1 MNPR

The source code of MNPR has been open sourced under the permissive *MIT License* and is stored at two different locations. For longevity and safekeeping, it has been uploaded to the data repository of the *Nanyang Technological University*, *DR-NTU* and is stored under the following DOI: 10.21979/N9/KU4B6S. For further collaborative development in the public domain, it has also been uploaded to a *Git* repository at *GitHub* under the following link: https://github.com/semontesdeoca/MNPR.

The source code has been well received within the research community and the animation industry, as well. The git repository has been forked more than 10 times within the first month of publication and more than 80 developers have starred the project. Two independent developers have already contributed to the source code, continuing the development in the public domain.

The raw substrate profile data (15x15cm) of 10 different papers and canvas, scanned at 2400 DPI using a *Canon LiDE 120*, was collected to extract the surface properties and create seamless textures thereof at two resolutions: 2048×2048 and 4096×4096 pixels. The seamless textures include the UV surface inclinations in the red U and green V channels, together with the heightmap in the blue channel. These were widely used to support substrate-dependent effects within the watercolor, charcoal and oil stylizations in MNPR. The raw scanned data and the seamless textures of the 10 substrate specimens have been uploaded to *DR-NTU* and are available to everyone under the same licensing conditions as MNPR. The substrate data is stored under the following DOI: 10.21979/N9/HI7GT7.

A class diagram of MNPR is also given in Figure A.1. This diagram can facilitate interested readers to visualize and interpret the source code for future use and contribution. While the *pluginMain* file in the source code registers the *Maya* render override, nodes and *mnpr* command, the stylization and all operations are defined in *mnpr_renderer* and its individual *style_* files. I would like to refer interested developers to the *Readme* file available at the *Git* repository for implementation details on how to integrate stylization pipelines to MNPR and take advantage of all levels of control.

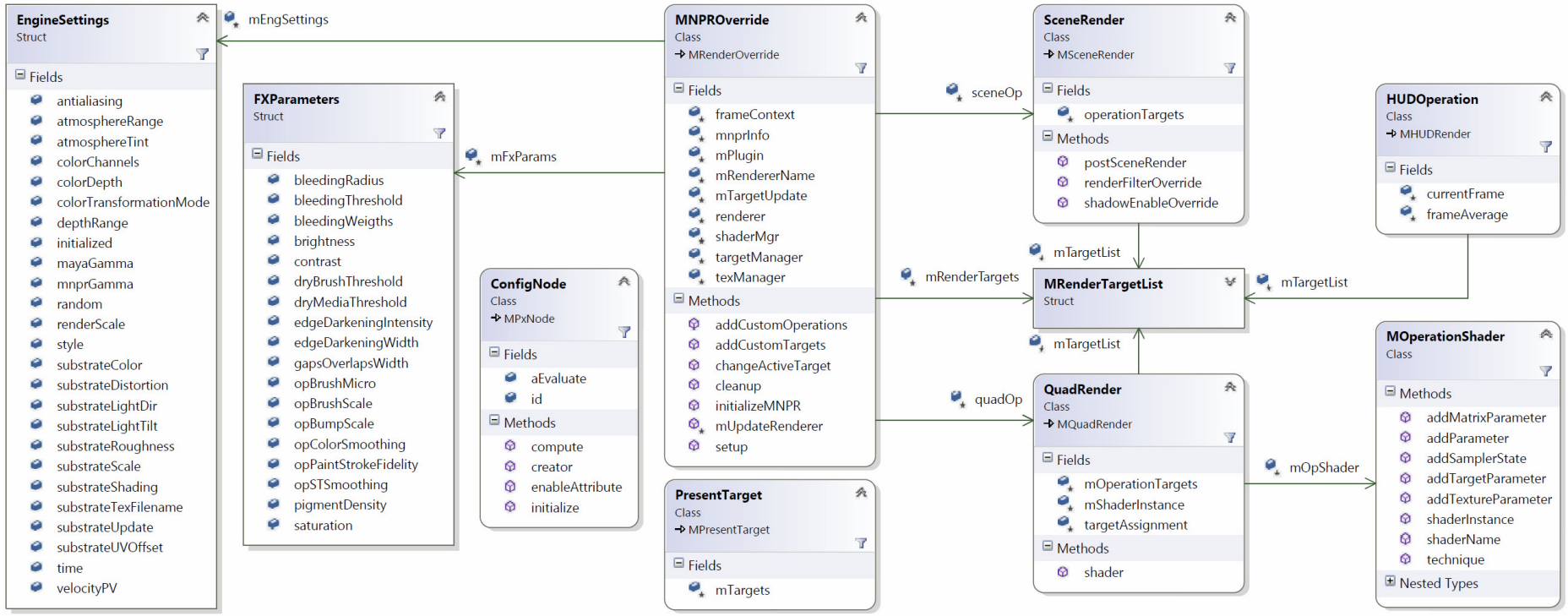


Figure A.1: Abstracted class diagram of MNPR. The *MNPROverride* class defines the render override and stylization pipeline, which can consist of *SceneRender*, *QuadRender* and *HUDOperation* operations. All different operations render to an *MRenderTargetList*, which was extended from the *Maya API*. The renderer is controlled through the *EngineSettings*, whereas the effect global parameters are set through the *FXParameters*.

A.2 User Studies

Two different user studies were conducted to evaluate/validate our contribution and gather feedback/insight from the target users: computer graphics artists. This section refers the reader to the datasets of both user studies, including the questionnaires and their responses, which have been stored in the data repository of the *Nanyang Technological University*, Singapore, *DR-NTU*.

The first conducted user study evaluated the reception of the watercolor stylization, the usefulness of a direct stylization pipeline and the needs from artists towards a better stylization. It involved five professional artists who watched a series of tutorials and tested the watercolor stylization with their own assets, to share their feedback and insights on the developed technology. The questionnaire and its responses may be found under the following DOI: 10.21979/N9/YHN5MW.

The second conducted user study evaluated the usefulness of the different levels of control within MNPR. It involved 20 artists and engineers from the animation industry, who watched a series of tutorials and tested the watercolor stylization with all the different levels of control, sharing their feedback and insights over their perceived usefulness. The questionnaire and its responses may be found under the following DOI: 10.21979/N9/XYZZVF.