# Real Time Rendering of Watercolor Stylized Character - Final Report

By: Hanna Keller

Supervised by: Prof. Mirela Ben-Chen

## Personal Motivation:

In winter of 2020 I studied the course computer graphics 1. Prof. Mirela Ben-Chen, my lecturer at the time, gave us a bonus assignment – investigate and find non-trivial NPR (non-photo realistic) techniques. When I searched the internet for such a technique, I came across the article "Art-directed Watercolor Stylization of 3D Animations in Real-time" by S. E. Montesdeoca, H. S. Seah, H.-M. Rall, and D. Benvenuti. The article discussed rendering of 3D models using openGL and C++ in the same manner as we learned in our graphics course, but with some advanced improvements that resulted with beautiful watercolor stylization. I love to draw, especially with watercolors, and the studies result inspired me. I searched some more and found out that the publishers of the article are selling artists a program that renders 3D models with watercolor stylization in real-time, and I decided that I want to make such a program myself as a personal project.

## Background:

Generally, the characters stylization for films and television is done in two steps. First, the characters images are rendered from a common 3D application, such as Blender or Maya. Then all the characters are gathered in a stylization program, in which the artists do the stylization of the characters and the scene.

The problem is that artists tend to alter the image a lot, but they can determine if the image needs changing only after they see the final result. If they decide that a change is required, they often need to go to the 3D applications and change some of the characters. But making stylization changes in the 3D applications without immediate feedback isn't intuitive, and in addition, the artists that work with the 3D applications aren't necessarily the same artists who work on the final scene, and they may have different visions and interpretations of the stylization.

All the above makes the stylization process slow, frustrating, and prone to several repetitions.

## Goal:

The target of this project is creating watercolor stylized images from triangulated mashes, by adding watercolor features to the image, such as color bleeding, hand tremor, pigment turbulence, etc. The project is artist-oriented, allowing the user to change the watercolor parameters and see the results in real time, thus giving the artist immediate feedback and saving the artist a lot of time.

## Method:

The program features a direct stylization pipeline that is art-directed in object-space by the end user, who can change the desired simulated effects with real-time performance. This enables immediate and coherent control over the watercolor look.

The program renders "classically" a 3D model from a triangulated mesh, and then adds some watercolor features to the "classical" model rendering. In addition, to "complete" the scene the user can add additional enrichment details to the scene such as wall, floor or skybox, and add a watercolor stylized background of their desire.

The watercolor features can be divided to several groups:

• Object-space shader with a watercolor reflectance model, art-directed pigment turbulence and density.

• Color blur and bleeding.

• Stylization shader with edge darkening, hand tremor, and additional color bleeding.

• Surface granulation with dedicated control over paper roughness and pigment accumulation at the valleys of the paper (art-directed).

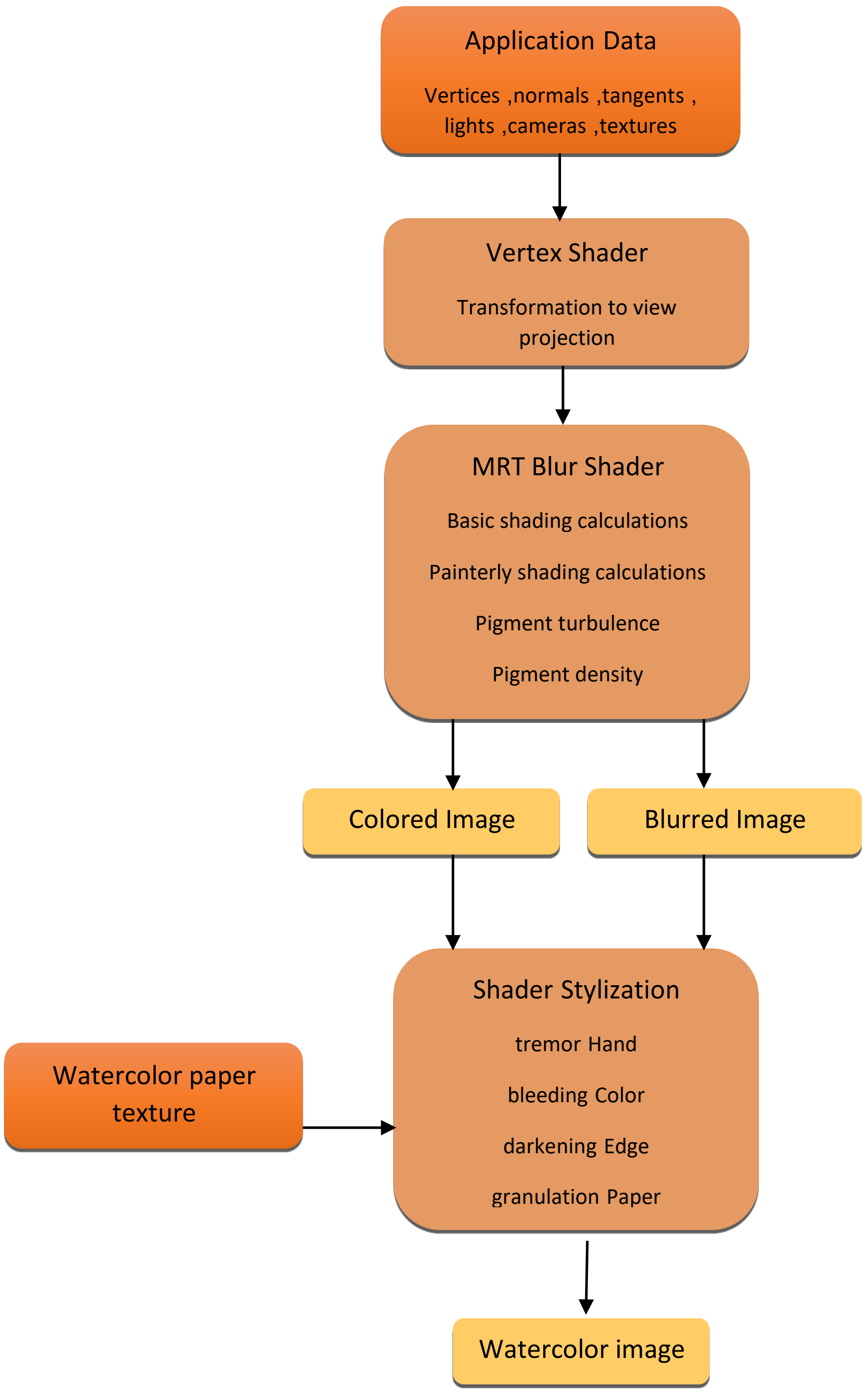

Fig.1. System schematic of the rendering pipeline. Dark orange elements represent input data, medium orange elements represent the different processing stages and the light orange elements represent the computed images in the frame buffer.

## Watercolor Features:

- *Hand Tremor Deformation –*
  Hand tremors is a very important characteristic of hand painted art, especially watercolor paintings. They are observable in the small irregularities found mostly on the edges of objects in a painting and are caused by involuntary fluctuations in the human nervous system.
  In the paper, to achieve the hand tremor effect, the authors translated the vertices, which contained embedded bleeding parameters in their vertex colors, along their normal vectors, and added sinusoidal noise function as it can additionally simulate watery deformation. Unfortunately, the equation that was given in the paper didn't work for me and gave very strange results. I felt frustrated, because I knew how important this effect is to the watercolor simulation.
  After several weeks in which I couldn't find the problem, I searched for a different implementation of the effect. I found a method that slightly moves the texture coordinates, making the illusion of tremor and color bleeding. I used it and I'm very happy with the result.

- *Watercolor Reflectance Model, Dilution and Pigment Turbulence –*
  Two of the most famous characteristics of watercolors are their transparency and turbulency, but those properties aren't part of the common shading primitives. The article proposed a custom watercolor reflectance model extending the common shading primitives: the shading model features watercolor characteristics such as pigment dilution, cangiante illumination (change in the highlights to a brighter hue) and art directed pigment turbulence.
  Initially, I simulated the **dilution** using the equation that was presented in the paper. But then I noticed that dilution - the "fading" effect - shifts the hue of the pixel to its background, which is not always white (for example in case the user used a skybox), so I modified the equation so it supports non-white backgrounds.
  I simulated the **cangiante illumination**, which shifts the hue of the surface away from a primary color using the equation that was presented in the paper.
  **Pigment turbulence** is a low-frequency noise, which happens when the pigment settles unevenly on the flat surface of the paper, creating areas of varying pigment density. This effect is characteristic for watercolors – due to the fluidity of the water – and is incorporated within the object-space shader.
  I simulated the **Pigment turbulence** using the equation that was presented in the paper, but for simplicity and lack of time, with constant noise texture (the artist can enlarge or decrease the significance of the noise in the model texture). In future versions this feature can be improved by using "natural noise" (not noise texture) and giving the artist control over its "levels".

- *Color Bleeding –*
  The Wet-in-Wet technique, often referred as "color bleeding", is a technique where pigment is applied to an already wet surface, to allow the color to spread (bleed) outside of its placement area.
  I already mentioned that the hand tremor effect caused small color bleeding. But to cause significant color bleed, I convolved a 5x5 Gaussian kernel over the entire color image and blended/masked the resulting low-pass image with the color image (using MRT shader). The authors of the paper added additional bleeding algorithm, which I considered too complicated and unnecessary for my program as I was happy with the bleeding effect I got with my solution.

- *Edge Darkening -*
  Edge darkening is a characteristic effect of watercolor that happens when a pigment is carried out to the boundaries of a colored area, due to surface tension. This causes a higher concentration of pigments at the edges of the image and therefore darker edges of colored areas.
  To model the gradient pigment accumulation found at the edges, the authors used the difference of Gaussians (DoG) feature enhancement algorithm.
  Unfortunately, the equation that was given in the paper didn't work for me and gave very strange results. After multiple unsuccessful attempts of using the given equation, I went back to one of the previous approaches of simulating edge darkening that was mentioned in the paper – the Sobol filter. Though it doesn't result with graduate darkening, it emphasizes and darkens the edges and I was satisfied with the result.

- *Paper Granulation -*
  Paper granulation is the result of the higher concentration of pigments found at the valleys of the paper, which generate a darkened appearance. In order to simulate both effects, actual paper texture was combined with the result image from the previous pipeline stages.

## Non-Watercolor Features:

Beside the watercolor features, I implemented some additional features in the program that help the artist create the full scene and adjust the watercolor stylization of the character.

- *Primitives –* the artist can choose (and switch in real time) whether to load a model or a primitive. It can be very useful – for example, one can get familiar with the watercolor stylization features and train on a primitive before diving into real model stylization.

- *Wall, floor, skybox –* all can be activated and transformed by the artist. Also the artist can load textures of his choice on each element, and perform watercolor stylization on the texture, thus creating perfect background for the model.

- _Lights_ – the artist can activate direct light, spotlight, and up to 4 point lights. He has control over the position/direction of the light, its color, etc. This setup allows the artist to set the illumination of the model as he sees fit.

- _Shadow_ – if the artist activates the direct light and the floor, for realistic reasons the shadow of the model appears. Watercolor stylization can be performed on the shadow.

- _Shading_ – when I did my research on watercolor rendering, I noticed that some papers recommended toon shading for better watercolor stylization effect. I gave the user 3 shading options: Grid (so the original geometry can be seen), Phong (classic) and Toon shading. When the toon shading is activated the user can choose the amount of color "levels".

- _Material_ – each model/primitive can be loaded with texture/material/both. The material properties can be customized by the artist. The model's texture is read automatically from the .mtl file in the objects folder.

- _Water_ – if the floor is activated, water animation can be activated on the floor.

- _Keyboard_ – each key in the keyboard acts differently on the program, which appears very handy when the artist is familiar with the program and wants to work quickly. The actions are described (and can be tuned or disabled) in the program's settings menu. Alternatively, all the actions can be done intuitively from the GUI.

And of course, all the elements (including the camera) can be transformed to all directions for easy scene construct.

## Future Work:

As much as I wanted to keep perfecting the program, my time was limited. I'll mention here few features/improvements that can be added to the program (for some of which I already built some foundations in the code):

Watercolor related features:

- _Turbulence_ – using of real noise (and giving the user control over its ambiguity) instead of constant noise texture.

- _Paper distortion_ – it’s a feature that was implemented in the paper but I couldn't make it work before I ran out of time. 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 of the paper texture.

- _Edge Darkening_ – using the difference of Gaussians (DoG) feature enhancement algorithm instead of (or in addition to) the sobol filter to achieve graduate edge darkening.

Non - Watercolor related features:

- *Normal and height maps* – add support to normal maps and height maps of the model (I built the foundations for it in the code but there were some issues I didn't resolve so it's commented out in the current code version).

- *Disable watercolor stylization* – add option to quickly disable all the watercolor stylization (right now you should zero all the watercolor parameters one by one).

- *Enable different skybox loading* – right now the skybox textures are hard coded in the code. Give the user option to load different skybox textures.

- *Add models* – give the user option to load concurrently several models and primitives (right now the program supports loading of one model at a time, but it can be constructed from several meshes).

- *Improve memory complexity* – right now every element added to the scene consumes a lot of memory and slows down the program (probably because of hidden memory leakage). Fix the memory leakage.

- *GUI: Load file using windows file explorer* - right now file are loaded by inserting their relative path in the GUI. It will be much more comfortable for the user to choose the file from a windows explorer window. I built the foundation for it in the code but encountered privileges issues that I had no time to resolve.

- *Installation wizard* – add an installation wizard to the program.

And of course there are a lot more cool features that can be added. For more fun ideas I recommend checking out the program that was implemented by the authors of the paper in the site:

https://artineering.io/publications/Art-Directed-Watercolor-Rendered-Animation/

## Bibliography:

[1] "Art-directed Watercolor Stylization of 3D Animations in Real-time" by S. E. Montesdeoca, H. S. Seah, H.-M. Rall, and D. Benvenuti.

[2] "Art-directed Watercolor Rendered Animation" by Montesdeoca SE, Seah HS, Rall HM.

## Results: