Interactive non-photorealistic rendering using GLSL

Charlotte Hoare

August 20, 2012

Contents

Chapter 1

Introduction

1.1 Non-photorealistic rendering

The goal in computer graphics for photorealism has been ever present. Photorealism, coming from the greek 'photo' meaning produced by light and realistic meaning depicting the real, is the term used to describe the pursuit of computed generated images so realistic they could be photographs of the real world. Naturally, many photorealistic techniques are physically based and simulate the real world. The most pertinent physical simulations being that of the interaction of light with objects and environment, and the projection of an image onto film using a camera. An example of a photorealistic rendering technique which uses these physical simuations is raytracing.

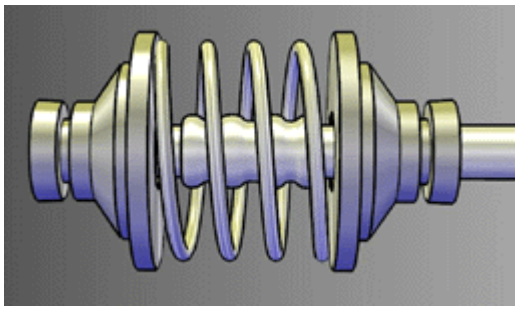

Figure 1.1: Gooch shading. An example of non-photorealistic rendering in technical illustration [10].

Photorealism exists beyond computer graphics. One example is in the work of the Dutch artist Vermeer, whose paintings can appear photographic in quality with both perspective closely observed, and brush strokes not easily discerned in the finished painting. However, his work was criticised for being inartistic and sterile and, this, along with the advent of photography, has led to photorealism in art being rare and more abstract forms of expression becoming more popular. These criticisms of photorealism in art have been equally levelled at photorealistic computer graphics, with images described as being too perfect and lacking in scope for artistic expression.

As such, research into alternative forms of artistic rendering has become popular; this research is generally referred to as non-photorealistic rendering. It is a many and varied topic, encompassing the emulation of traditional art forms, technical illustration and many more. Where photorealism is concerned with simulating the behavior of light, non-photorealistic rendering is driven by the more subjective human perception. This is very difficult as where physically-based simulation is governed by replicable equations, it is not easy to replicate the subjective processes of an artist. Given this, it is necessary to understand the creative process of the artist as well as the physical aspects to be emulated.

Non-photorealistic rendering brings the disciplines of art and science together; its value lies in both its techniques and, importantly, on the aesthetics of the results. It is this joining of art and science which is the most exciting aspect of non-photorealistic rendering.

1.2 Watercolour rendering

One area of non-photorealistic rendering is painterly rendering. The advantages of painting stem from the fact that by not depicting every detail, the painter allows the viewer to complete the picture and thus share in the interpretative process. Differing brush strokes can define the character of a surface, that is, how light is reflected from it; surfaces with smooth blending brush strokes imply smoothness or softness where harsh, unblended strokes imply stronger lighting or more pronounced surface texture. In other words, a painting can provide focus or improve

composition by abstractly inviting the viewer into the interpretive process. It is this abstraction which is the aim of painterly rendering.

Painterly rendering techniques are varied, some using particle systems and others point sprites. Meier suggests using particles to describe a surface and then render each particle as two-dimensional brush strokes, whilst Sperl renders the particles as point sprites. However, these methods can also suffer from looking sterile; the particles or point sprites are too uniformly distributed and the brush strokes for each unvarying. For this reason, the focus of this thesis changed toward a painterly rendering style which both simulates physical phenomena as well as incorporating the creative process in order to get a more aesthetically pleasing look; watercolour rendering.

Watercolour painting offers a very rich medium for artistic expression. Its main features stem from the variation of pigment saturation, the behaviour of the water and the visibility and texture of the paper. These features result from the interaction between the media, water, pigment and paper, and can therefore be scientifically described, observed and emulated, whilst retaining the creative scope essential to non-photorealistic rendering. In other words, by emulating watercolour effects, rather than simulating them, it is possible to present a set of intuitive controls to the artist which will be the aim of the program.

1.3 Ink rendering

Another area of non-photorealistic rendering addressed in this project is ink rendering. In the real world, pen-and-ink line drawing techniques are often used to great effect to depict form, tone, and texture in artistic, technical, and scientific illustration. Its broad employment across these genres of illustration stems from the fact that it essentially clarifies shapes and focuses attention. As such, non-photorealistic rendering in this style has been the subject of much research.

One particular, more artistic, case of pen-and-ink rendering is that of Chinese ink rendering. Chinese ink painting is very abstract, and incorporates "implicit meaning" in which painters use a minimum amount of brush strokes to express form. For the purposes of this thesis, this more artistic ink renderer is pursued; with large brush strokes similar to Chinese ink painting. That is, a method to automatically draw 3D models in the style of Chinese ink painting is sought. This is in line with the end-program being an intuitive tool for artists, as discussed in the previous section.

1.4 Structure of the thesis

The thesis is structured as follows. In Chapter 2, the previous work relating the problems to be addressed is discussed. This includes a survey of relavent previous works, highlighting any particularly pertinent points to be noted. Chapter 3 is concerned with the implementation of the program, including an overview of utility methods employed as well as descriptions of specific non-photorealistic rendering algorithms. In Chapter 4, the results of the program, and the project in general, are detailed. Also presented here are concluding thoughts and remarks on the project.

Chapter 2

Previous Work

2.1 General

As mentioned in Chapter 1, interest in non-photorealistic rendering research has steadily grown in the last decade. As such, there are ample books classifying non-photorealistic techniques and are the best places to begin research. In the case of this thesis, the reference guide used was "Non-Photorealistic Rendering" by Bruce and Amy Gooch [1]. This book is one of the major authorities on non-photorealistic rendering. It provides an overview of the published research on non-photorealistic rendering and categorises the current research into an encyclopedia of usable algorithms and techniques. The most important points to note from the book are concerned with forming an idea of what non-photorealistic rendering is and what it should achieve, namely that "simulating reality is not as important as creating the illusion of reality".

2.2 Watercolour rendering

Curtis et al. in 1997 [2] described the various features of watercolour painting, and showed how these effects could be simulated. The salient points of this paper are the categorised properties of watercolour painting and not the simulation, as the aim of this project is to emulate the features of watercolour painting with a view to creating intuitive tools for the artist.

Before continuing, it is necessary to define some terms related to watercolour painting which will be frequently used in this thesis, and which are also categorised in [2]. Watercolour paintings are created by the application of watercolour paint to paper. *Watercolour paper* is different from other paper in that it is made from linen pounded into small fibers. The paper itself is mostly air, laced with a microscopic web of these tangled fibers. As such, watercolour paper is extremely absorbent to liquids, and so the paper is impregnated with something called *sizing* so that liquid paints may be used on it without immediately soaking in and diffusing. Sizing, made from cellulose, forms a barrier that slows the rate of water absorption and diffusion.

Watercolour paint is, in essence, a suspension of *pigment* particles in a solution of water, and chemicals called binder and surfactant. A pigment is a solid material in the form of small, separate particles which hold the colour of the watercolour paint and penetrate into the paper. Once they have settled in the paper they tend not to migrate. Pigments vary in density, with lighter pigments stayin suspended in water longer, and, as a result, spreading further across paper as the water travels. The two other chemicals in watercolour paint, binder and surfactant, also have important roles; the binder enables the pigment to absorb into the paper, while the surfactant allows water to soak into paper impregnated with sizing.

Given these definitions, it is possible to explicitly define some of the main features of watercolour painting. The main properties described include edge darkening, which can be seen in Figure 2.1. In a watercolour brushstroke, the amount of sizing in the paper together with the surface tension of the water means the brushstroke is prevented from spreading. As a result of this, the pigment gradually migrates from the interior of the painted region towards its edges as the paint dries, which leaves a dark deposit at the edge of the brushstroke. This edge darkening is a key effect and one that watercolour artists utilise in their expression and one that paint manufacturers continue to ensure is part of their watercolour paint formulations.

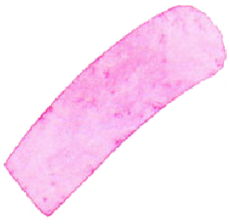

Figure 2.1: Edge darkening [2].

Figure 2.2: Granulation due to heavy-density pigment [2].

Another pertinent feature described in the paper is that of the granulation and separation of pigments. The granulation of pigments results in a grainy texture which emphasizes the gradient of the paper as the watercolour runs into the valleys of the paper. It is similar to separation; a splitting of colours that occurs when denser pigments settle earlier than lighter ones. Combined, this results in pigments with a greater density showing more of the effects of granulation, which can be seen in Figure 2.2. These features are the salient ones when it comes to watercolour painting, and as such need to be properly emulated in the final application.

Another paper which describes attempts non-photorealistic watercolour rendering is [3]. Not only this, it also attempts to emulate watercolour effects rather than simulating them to offer intuitive controls to an artist, in line with the aims of this thesis. On top of those effects covered in [2], it defines two more, highlighted below.

The water component of watercolour paint causes colour variation due to the non-homogeneous distribution of water on the canvas. The flow of water is turbulent and so this effect shall be referred to as turbulence flow. An example of this in a real watercolour can be seen in Figure 2.3. In addition to this, a wobbling effect is discernable in watercolour paintings due to paper grain; that is, as a result of pigments depositing in small valleys in the paper. This can also be seen in Figure 2.3.

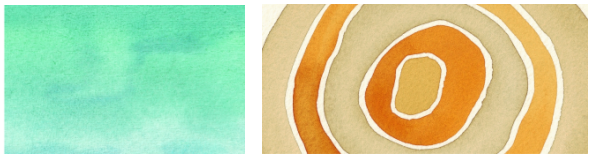

Figure 2.3: Turbulence flow and wobbling [3].

Another important paper in watercolour rendering is [4], which combines the simulation and emulation elements of the aforementioned papers with interesting results, and also [5] which applies the theory in [3] as an image processing technique; an interesting possible extension.

2.3 Ink rendering

[6] provided the best overview of both Chinese painting expression integrated with computer technology. It suggests that Chinese ink painting is all about implicit meaning, that is, a minimum amount of strokes is used to express meaning and emotion. This is very similar to silhouette shading (see Figure 2.4); a common non-photorealistic technique on which there are many resources [7].

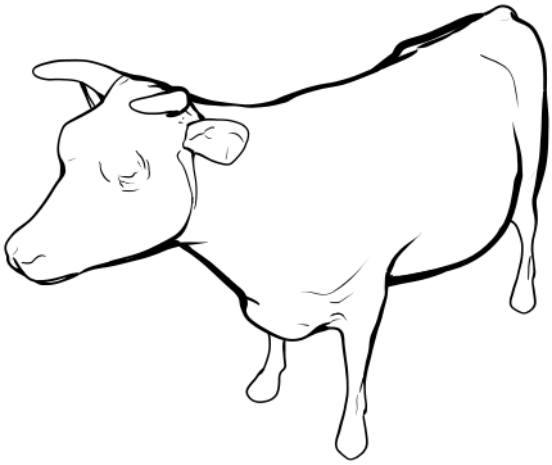

Figure 2.4: Silhouette shading [7].

Chapter 3

Implementation

The problem then to be solved, which can be explicitly stated following Chapters 1 and 2, is this: Given a 3D scene, render the scene as a watercolour painted image or an ink painted image. For the purposes of this chapter, we will use a test scene containing a golden teapot, which can be seen in Figure 3.1. As proposed in [3], the first step is to apply cel shading to the scene, and then apply the watercolour effects to the cel shaded scene.

Figure 3.1: Test scene.

3.1 Cel shading

Cel shading is a type of non-photorealistic rendering specifically designed so that 3D computer graphics appear hand drawn. Indeed, its name is derived from cels, the clear sheets of acetate which are painted on in tradition 2D animation. Often used to mimic the style of a comic book or cartoon, its most common usage is in video games.

The algorithm used to implement cel shading in the application is a simple one, which performs light intensity thresholding in order to create four discrete surface shades which are lighter where the threshold is high, and was written based on the tutorial at [8].

As light intensity is strongest when the directon of light n the scene and the normal at the surface are coincident, and drops off to zero as the angle between the two increases to 90° [9], the light intensity on a surface is equivalent to the cosine of the angle between the light direction vector and surface normal. That is,

$$\begin{aligned} \textit{intensity} &= \cos(\mathbf{lightDirection}, \mathbf{normal}) \\ &= \frac{\mathbf{lightDirection} \cdot \mathbf{normal}}{|\mathbf{lightDirection}| * |\mathbf{normal}|} \end{aligned}$$

This can be simplified if both the direction of the light and the normal are normalised (they have length one) so that,

$$\begin{aligned} \textit{intensity} &= \cos(\mathbf{lightDirection}, \mathbf{normal}) \\ &= \mathbf{lightDirection} \cdot \mathbf{normal} \end{aligned}$$

This calculation may be performed using GLSL's "dot" function, so the light intensity becomes trivial to find per vertex or per fragment.

In this case, the light intensity is found per fragment in the fragment shader. The vertex shader normalises the vertex normal and then passes this to the fragment shader for intensity calculations, then computes the homogeneous vertex position. The fragment shader is more involved. It takes in the normalised normal from the vertex shader and also a uniform colour from the application (see Figure 3.2 for the colour passed in in the test scene). The direction of the light is specified and normalised, and then the light intensity per fragment is calculated. The intensity thresholding is then performed. Where the intensity is greater than 0.95, the uniform colour value passed in is outputted as the fragment colour. Otherwise, if the intensity is greater than 0.5, the uniform colour is scaled down and outputted as the fragment colour, and so on. The result of the intensity threshold cel shader applied to the test scene can be seen in Figure 3.2.

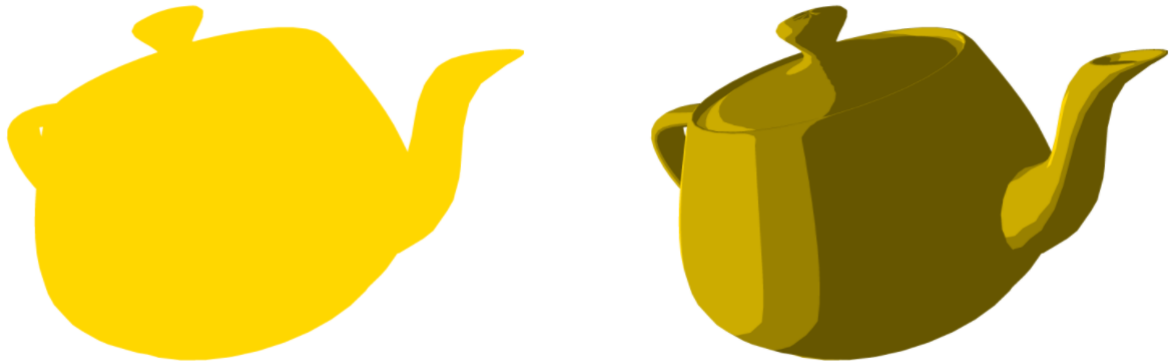

Figure 3.2: Test scene with cel shading.

3.2 Framebuffer objects

Many of the watercolour effects defined in Chapter 2 need to be applied to the entire screenspace, as it is this which is analogous to the painting in the application. In order to apply effects to the entire screenspace, the frame buffer is captured as a frame buffer object and stored in a texture. This texture then sent to a GLSL shader as a sampler, and this shader is then used on a plane which covers the OpenGL buffer which is seen in the final application.

The GLSL vertex and fragment shaders which are used to shade the plane, which is now essentially the analogous painting, are where the rest of the watercolour effects will be implemented. The vertex shader passes the uv co-ordinate information to the fragment shader as **vertUV** and then calculates the homogeneous vertex position. Again, the fragment shader is more involved.

Figure 3.3: Test scene with wobbling.

3.3 Wobbling

When wet watercolor paint is applied on dry paper, there is a wobbling effect due to the raggedness of the paper. This effect is emulated in the fragment shader by distorting the incoming framebuffer texture. This distortion is achieved by offsetting the uv co-ordinates using a paper texture passed in from the main program according to, in GLSlang,

$$\mathbf{wobbleUV} = \mathbf{vertUV} + texture(paperTex, \mathbf{vertUV}) * D$$

where $paperTex$ is the texture passed as a uniform sampler to the fragment shader and D scales the offset. The higher D is, the greater the distortion. This distortion scale factor is one of the intuitive controls which is made available to the user. To see the effect of the new wobbleUVs, see Figure 3.3.

3.4 Edge darkening

Before discussing edge darkening, it is necessary to describe the way colour darkening is achieved in the fragment shader, as it will be referred to here and in subsequent sections.

The darkening method used is that used in [3], where an empirical darkening due to pigment density model was described. It uses a parameter d to specifiy the pigment density, and the darkening is then computed according to the following equation,

$$C = C(1-(1-C)(d-1)) \tag{3.1}$$

Using a parameter d to control the darkening, or pigment density, is useful as it can be made open to the user.

Figure 3.4: Test scene with edge darkening.

In order to darken only the edges, however, it is necessary to first detect those edges in the fragment shader. Edge detection is based on the assumption that an edge occurs where a sudden change in colour occurs. Basic edge detection uses the change in intensity values, rather than colour values, to detect an edge. Here intensity is not the same as the light intensity described in Section 3.1 but rather colour intensity. The intensity of a colour is the brightness or force of a colour. In the fragment shader, the colour intensity of a pixel is found by averaging the value of the three colour channels of the pixel. The edge detection itself is implemented in the function 'edgeDetect' in the fragment shader and works as follows. The pixel intensity is found for the current pixel and its eight direct neighbours. The differences in the intensities of the neighbour pixels is then calculated and averaged. If this average is very small, that is, the differences in the intensities is negligible, the function returns that there is no egde. If the average is larger, then there is an edge.

To edge darkening. Essentially, for every pixel, if it does not lie on an edge, the function returns 0.0, if it does lie on an edge, it returns a nonzero float. When a nonzero value is returned in the fragment shader, that fragment is darkened according to equation (3.1).

3.5 Turbulence flow and pigment dispersion

As described in Chapter 2, the colouring of a watercolour varies in two ways; there is granulation causing pigment dispersion and non-homogeneous repartition of colour due to the turbulent flow of water. These colour variations are emulated in layers using greyscaled images sent to the fragment shader as uniform samplers. For the pigment dispersion layer, a greyscaled paper texture was again used and for the turbulent flow a Perlin turbulent noise texture was used. The intensity of these images then gives the colour variation. This method was based on one presented in [3].

When the greyscaled image has an intensity of I (as described in Section 3.4), the fragment is darkened according to equation (3.1) with a d value of

$$d = 1 + \beta(I - 0.5)$$

where β is a global scaling factor which can be made open to the user. The greater β, the more turbulent the flow of the water; the greater the granulation of the paper.

Finally, it is important that the fragments are only darkened where there has been paint; that is, the paper itself would not be darkened where it is blank. As such, the fragment shader first calculates if the raw colour information is fully white; that is it is paper. Then, later, if the raw colour information was found to be fully white, the turbulent flow and pigment dispersion is not applied. To see the final result of these colour varying effects, see Figure 3.5.

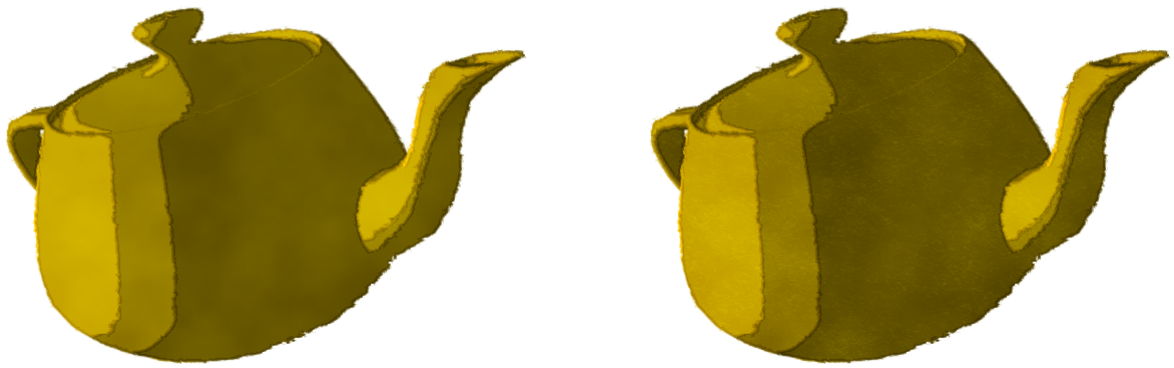

Figure 3.5: Test scene with turbulent flow on the left and turbulent flow plus pigment dispersion on the right.

3.6 Paper normal mapping

The final step in generating a watercolour from a 3D model was to make the plane appear bumpy like watercolour paper by using a watercolour paper normal map. The calculation is a simple one, the fragment shader looks up and sets the surface normal according to the watercolour normal map which has been passed to the fragment shader as a sampler. Following this, the diffuse lighting value is calculated in the normal way [9]

$$diffuse = \max((lightDirection \cdot normal), 0.0)$$

and then multiplied with the output colour to get a normal mapped plane which appears like watercolour paper. The final output can be seen in Figure 3.6.

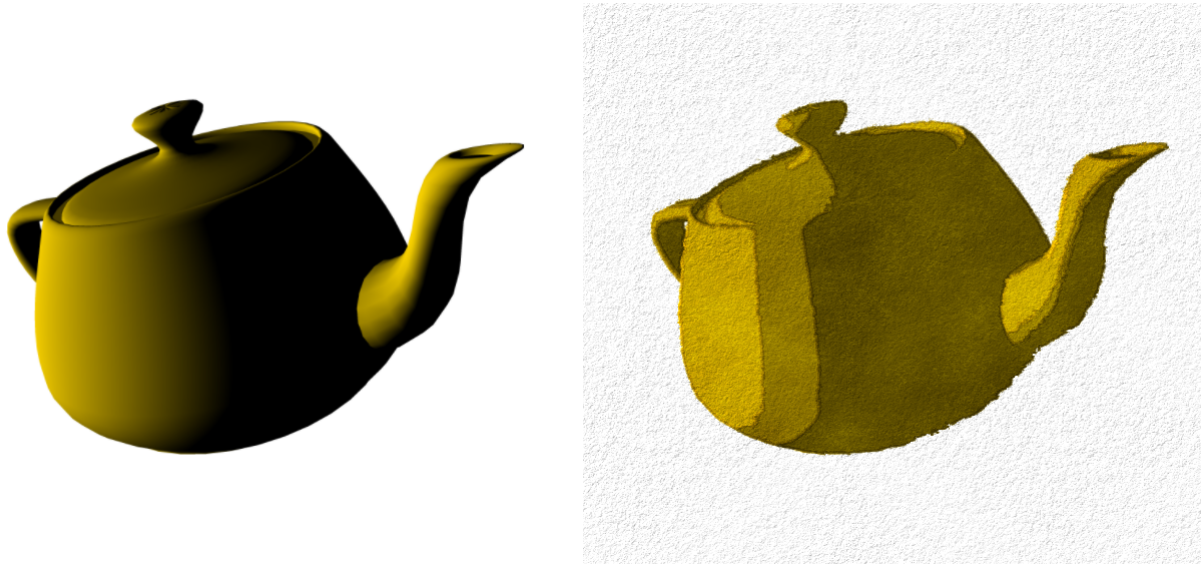

Figure 3.6: The final output compared with the original 3D model.

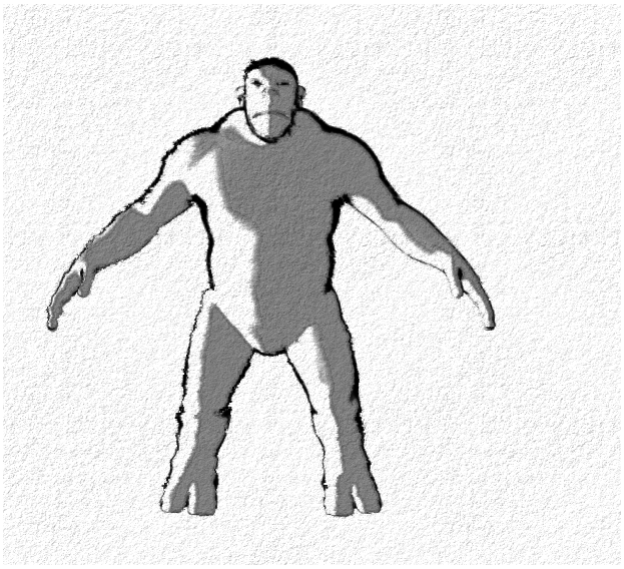

Figure 3.7: Troll rendered ink style.

3.7 Ink shading

The ink renderer is only slightly different, in that it implements ink shading where the watercolour renderer implements cel shading. The shader was written with help from [11]. The ink shader works by calculating light intensity like the cel shader, but the calculation is slightly different. The intensity is offset so that the places which form the silhouette of the model appear to be in the dark; the sihouette will render darkest. The interior works with the light intensity as normal, growing from dark grey to white to show intensity. The results of this shader can be seen in Figure 3.7.

Chapter 4

Conclusion

The resulting images from the program are visually very pleasing, as can be seen in Figure 4.1. The watercolour effects are convincing, and all are interactively controllable by the user and can be intuitively understood. To this end then, the project has been a success.

Some definite improvements would be to achieve some kind of real-time temporal coherence, in order to reduce the "shower door" effect of the turbulent flow and pigment dispersion effects. The "shower door" effect coming from the watercolour paper is actually quite pleasing, as it appears the movement is happening on a static piece of paper; the watercolour paper becoming a viewport. The excursion into non-photorealistic rendering and GLSL has been an interesting one, and certainly the knowledge and understanding gained in terms of GLSL is invaluable.

Figure 4.1: Resulting images

Bibliography

[1] Gooch, B. and Gooch, A., 2001. *Non-photorealistic Rendering*. Massachusetts: A K Peters.

[2] Curtis, C. J., Anderson, S. E., Seims, J. E., Fleischer, K. W. and Salesin, D. H., 1997. Computer-generated watercolor. *In: Siggraph '97*, ACM Press, 421-430.

[3] Bousseau, A., Kaplan, M., Thollot, J. and Sillion, F. X., 2006. Interactive watercolor rendering with temporal coherence and abstraction. *In: NPAR '06*, ACM Press, 141-149.

[4] Lei, E. and Chang, C., 2004. Real-time rendering of watercolor effects for virtual environments. *In: PCM '04*, ACM Press, 474-481.

[5] Doran, P., 2010. *Expressive rendering with watercolor*. Thesis, (Masters). Brown University.

[6] Way, D. L., Lin, Y. R. and Shih, Z. C., 2002. Chinese ink rendering for trees using outline drawing and texture strokes. *In: IWAIT '02*.

[7] Goodwin, T., Vollick, I. and Hertzmann, A., 2007. Isophote distance: a shading approach to artistic stroke thickness. *In: NPAR '07*, ACM Press, 53-62.

[8] Lighthouse3d.com. *Toon Shading*. Available from: http://www.lighthouse3d.com/tutorials/glsl-tutorial/toon-shading/ [Accessed 17 August 2010]

[9] Rost, R. J., 2006. *OpenGL Shading Language*. Second. Addison Wesley Professional.

[10] Reynolds, C., 2004. *Stylized Depiction in Computer Graphics*. Craig Reynolds. Available from: http://www.red3d.com/cwr/images/npr/gooch_98.gif [Accessed 17 August 2012]

[11] 2011. *Outline via GLSL?*. Maratis 3D. Available from: http://forum.maratis3d.com/viewtopic.php?id=56[Accessed 17 August 2012]